**D424 – Software Engineering**

**Task 3**

|  |  |
| --- | --- |
| **Capstone Proposal Project Name:** | http://www.idevnews.com/views/images/uploads/general/wgu_logo.png  Order Management Web Application |
| **Student Name:** | Adam Barham 012306215 |

**Table of Contents**

Table of Contents

[Table of Contents 2](#_Toc200442880)

[Task 3 Design Document 3](#_Toc200442881)

[Application Design and Testing 3](#_Toc200442882)

[Class Design 4](#_Toc200442883)

[ERD Diagram 5](#_Toc200442884)

[UI Design 6](#_Toc200442885)

[Unit Test Plan 7](#_Toc200442886)

[Introduction 7](#_Toc200442887)

[Purpose 7](#_Toc200442888)

[Overview 7](#_Toc200442889)

[Test Plan 8](#_Toc200442890)

[Items 8](#_Toc200442891)

[Features 8](#_Toc200442892)

[Deliverables 8](#_Toc200442893)

[Tasks 9](#_Toc200442894)

[Needs 9](#_Toc200442895)

[Pass/Fail Criteria 10](#_Toc200442896)

**Application Set-up**……………………………………………………………………..16

**User Guide**………………………………………………………………………………18

**Task 3 Design Document**

# Application Design and Testing

**Class Design**

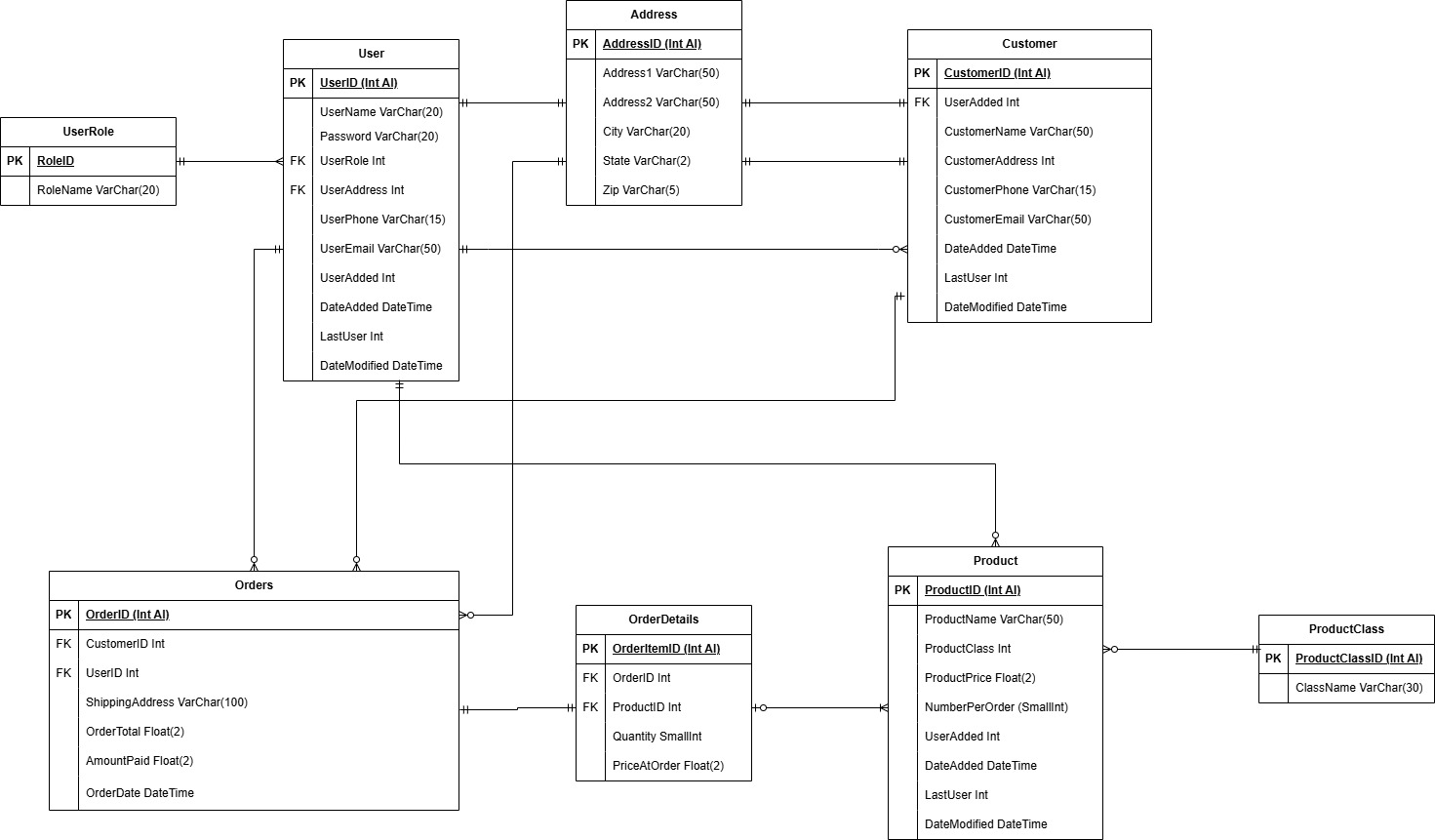
In this class diagram, the relationships and structure of data components are exhibited for a Customer and Order Management application build using ASP.NET and hosted on Microsoft Azure. This class diagram includes inherited classes User and Customer from the PersonBase class, as well as an Address structure, which is an owned entity in EFCore and does not receive its own database table.

The application is designed to manage customers and create sales orders for a small-medium sized electronics manufacturing company. The main functions in the application are handled through EFCore and server-side Blazor, which integrates the controller of MVC model into the code block of the page by allowing C# code to integrate with HTML objects. Thus, each page is a representation of a function, and classes are designed to mimic database structures.

By using server-side rendering with Blazor, the application can update in real-time with concurrent users, loads efficiently as the page is rendered server-side before being piped to the browser through a SignalR connection. This allows us to leverage EFCore to its maximum potential by writing functional code within each page. Included with the class diagram is an ERD diagram to show database parallels.

## Class Design

## ERD Diagram



## UI Design

The UI design of this project provides a mobile-responsive, sales-first approach for users, giving immediate access to the order form, monthly order metrics, and secondary functions. Pages are role-responsive, blocking access for elevated functions to base users. Blazor’s server-side UI updates in real time as data changes using a SignalR connection to render the page.

Users are immediately directed to a login page where they can log into the application. After logging in, the application moves to a dashboard page with month-to-date orders and a prominent feature to take users to order management. Order management, and creating a new order, allows users to create a new customer from that page without having to first navigate to the customers page, and each page allows users to return to the dashboard quickly.

In addition to managing users, products, customers and orders, a reports page is available with 4 important and relevant reports including customer balances for invoicing and product sales tracking. Also included is a migration tool, which allows users to upload a properly formatted .csv file to be integrated into the database, allowing the company to transition from excel based order management to the new web application tool without losing data.

The application is designed to be scaled up, with CSV exports for reports that can be easily integrated into an API for an invoicing and payment expansion, role based authentication that can be expanded for different business roles using Blazor’s AuthorizeView functionality, and easily expandable integration with database functionality.

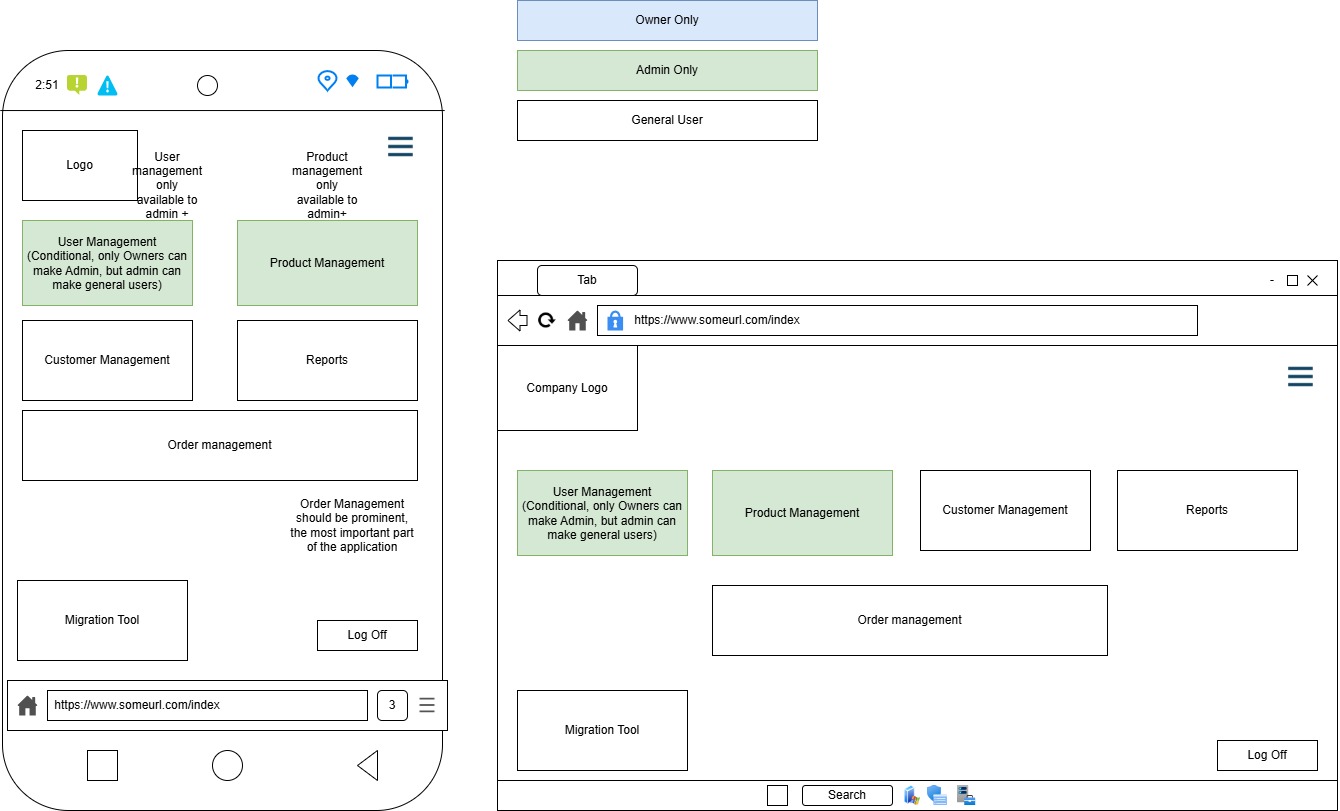
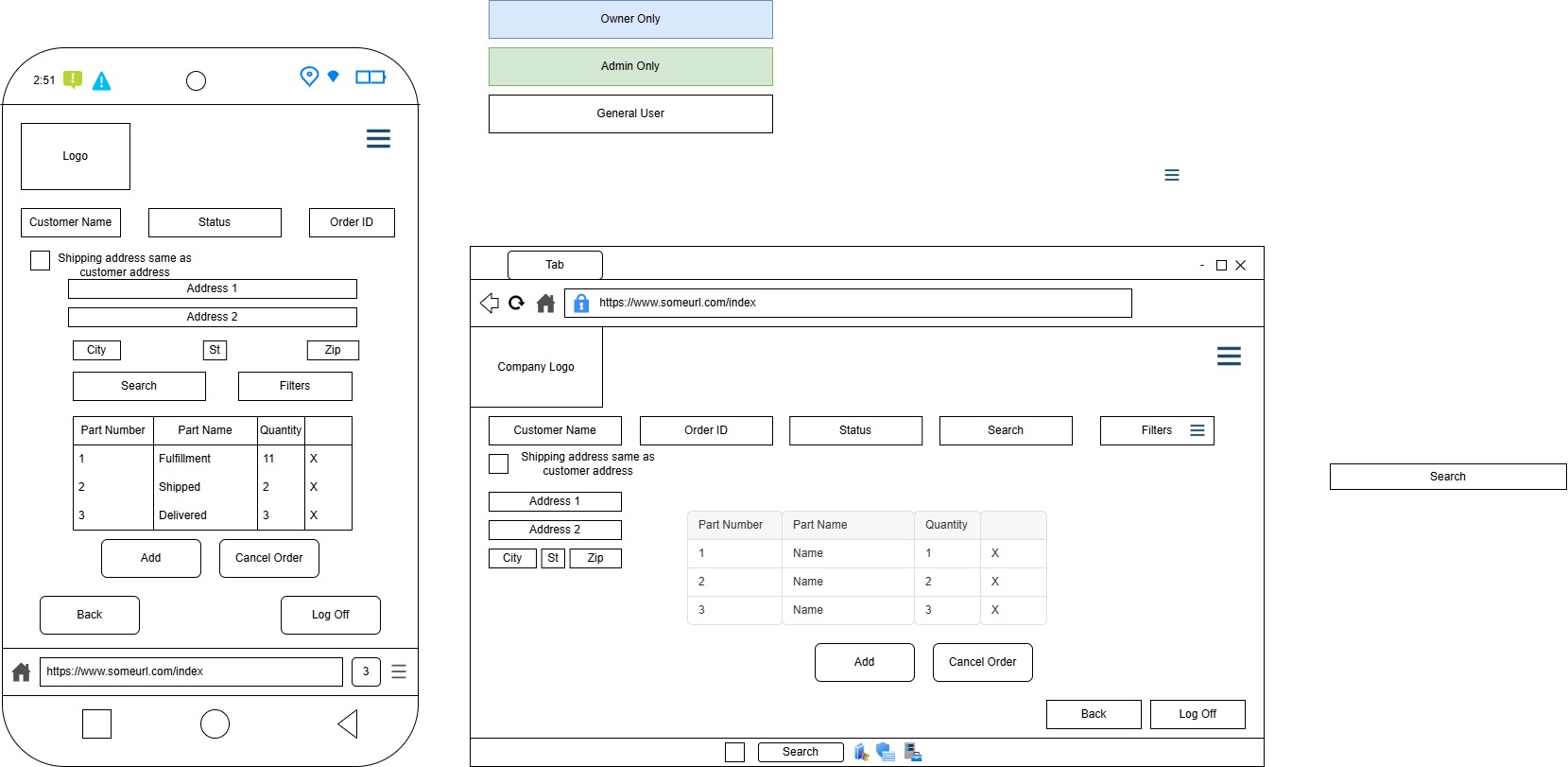
Figure 1: Dashboard

Figure 2: Order Page

# 

# Unit Test Plan

## Introduction

### Purpose

The purpose of the unit test plan is to ensure reliability of individual code components of the application. Isolation testing was used to verify functionality of individual components including database communication and password hashing, which allowed for immediate identification and addressing of defects.

### Overview

The unit test plan was used during the development process to validate unique or critical components of the application. Particular focus was paid to authentication and roles, verifying security and role-based permissions, as this has a cascading effect on the entirety of the application.

User Authentication: Tested user login function to verify successful user login, logout, requires password reset, and cookie-based authentication standards. When errors were encountered, including a failure to save the authentication cookie to the browser, code adjustments were made including writing a custom authentication state provider.

Database Interaction: Tested database storage and data validations using a test product and checking the changes against the database to ensure proper read/write functions. Any errors encountered during this test were explored as invalid data, database connection, or database constraint errors and code was adjusted accordingly to ensure proper communication.

## Test Plan

### Items

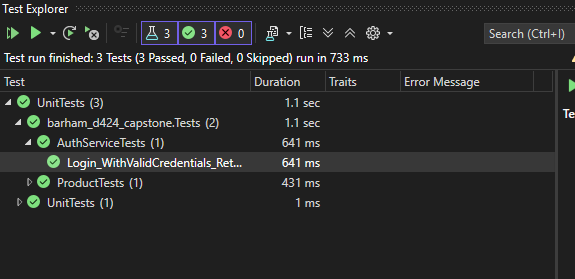
* Development Environment: Visual studio 2022, using .NET framework 8.0 (long term support).
* Database: AzureSQL database was established and pre-filled with test data by using simple SQL scripts in SQL Server Management Studio before writing application.
* Test Framework: .NET xUnit Test Project with EFCore.InMemory in order to prevent production database alteration from testing.

### Features

* User Authentication Test: Tests user authentication functions and access control
* Add Product Test: Tests the ability to add products to the database

### Deliverables

* Test scripts are included in the /UnitTests project included with the solution. Tests are written in C# using the .NET xUnit testing framework.
* Test results are included with the screenshot below.



### Tasks

* Set up IDE with required dependencies, and establish primary application project and test project.
* Write test scripts for each feature.
* Include test data in scripts to test features.
* Execute tests in IDE.
* Analyze results and make appropriate adjustments as needed.

### Needs

* ASP.NET, EFCore, and Xunit are the required package dependencies for these tests, which allows for AspNetCore’s authentication and http services, as well as database manipulation with EFCore.
* Access to application, provided by using a single solution for the application and the unit tests, and applying the appropriate dependency statement.

### Pass/Fail Criteria

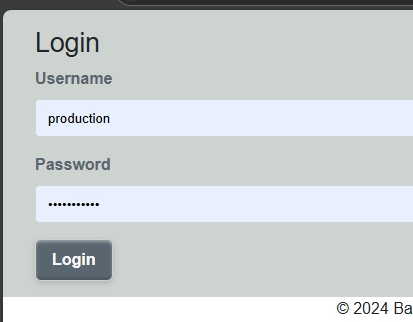
Unit Tests:

Two unit tests were written to verify functionality of Authentication service and product addition to the database. Unit tests were written in C# in the solution in a separate project and can be found under /UnitTests.

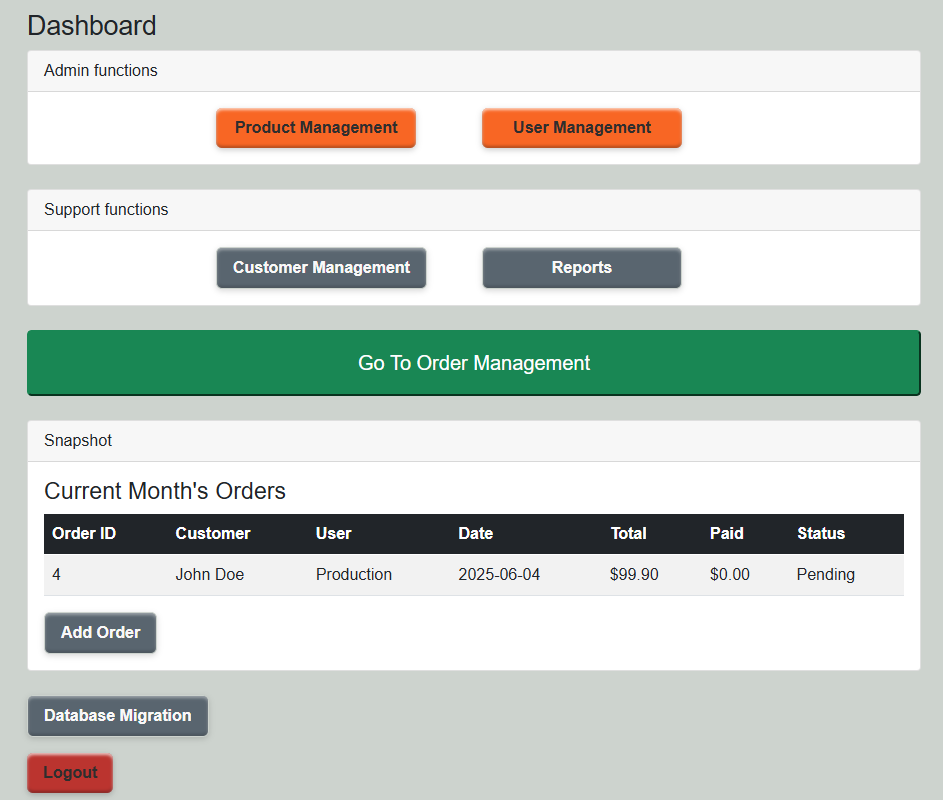
|  |  |  |  |
| --- | --- | --- | --- |
| Test | What is Expected | What Happened | Pass? |
| User logs in with correct username/password combo | The application successfully logs in and displays an appropriate page based on user roles | User logged in successfully, role based permissions in place, password reset functionality confirmed, and failed login confirmed | Yes |
| A product is added to the database | The database will show an additional product with appropriate information in all fields | Adding a product is confirmed and field validation is in place | Yes |

Screenshots of additional functional testing:

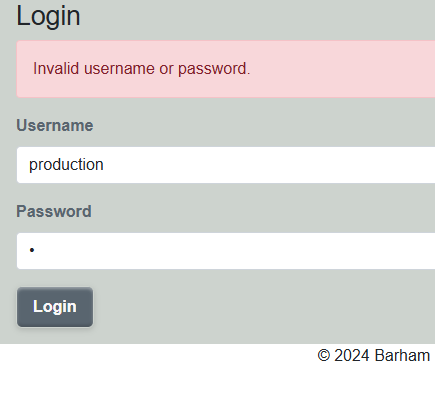
Login with correct credentials, Owner+ role:



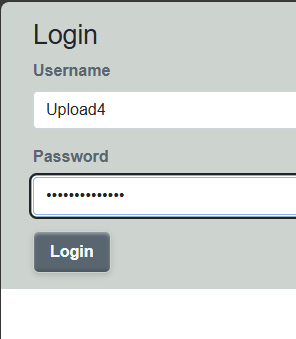
Correct page with Admin functions:



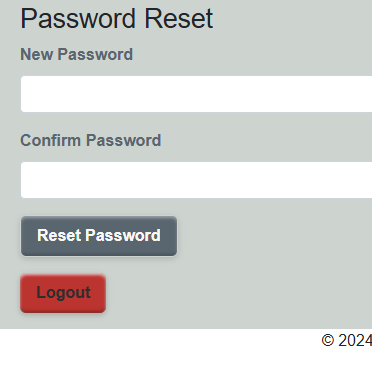
Invalid login:



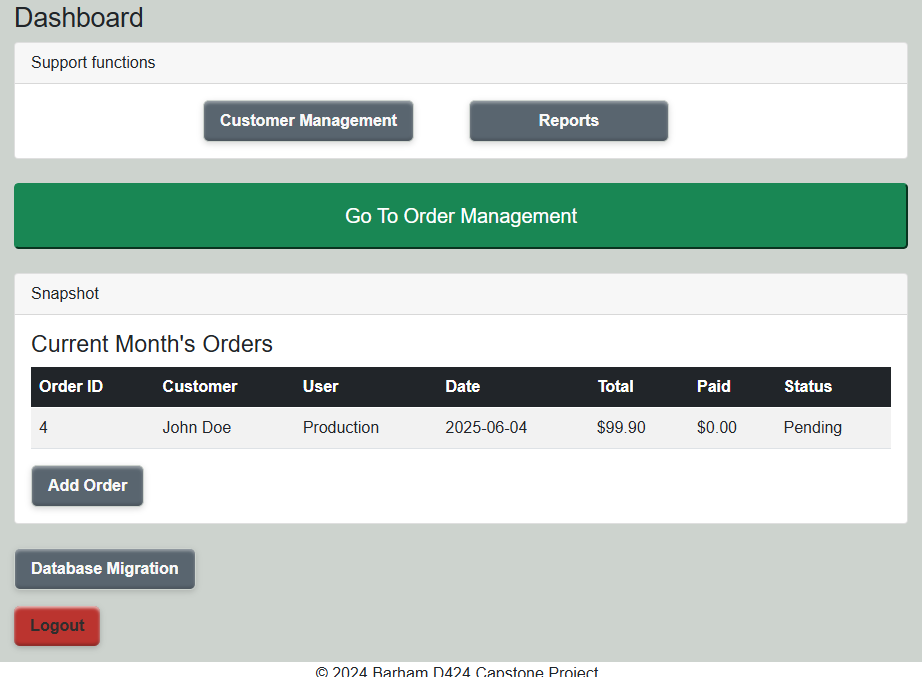
Login with new user requiring a password reset:



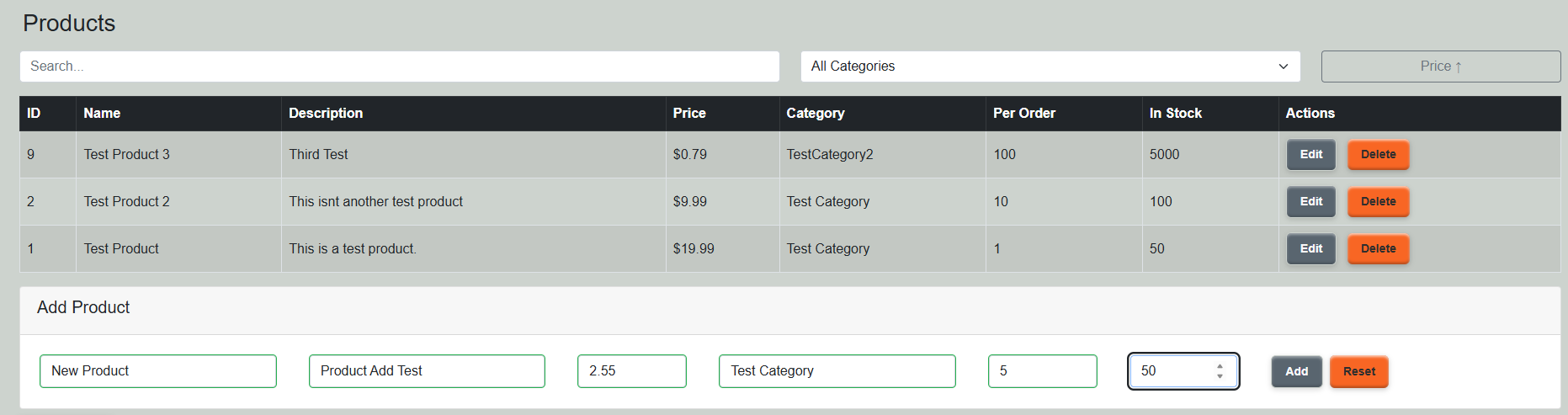
Requires password reset:



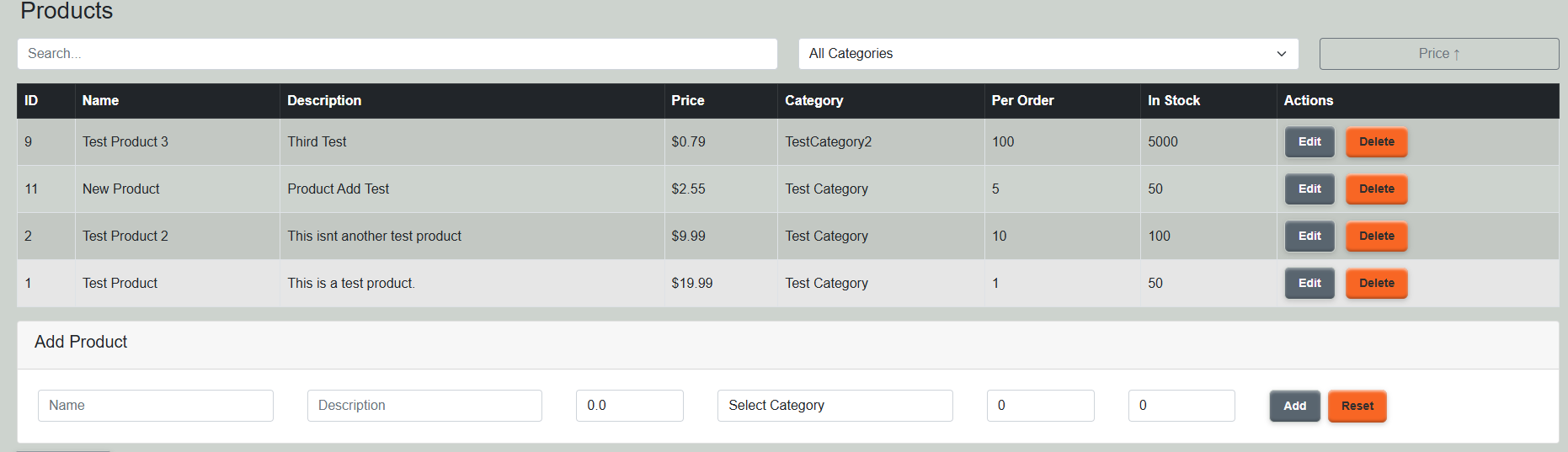
User logs in without admin access, admin features blocked



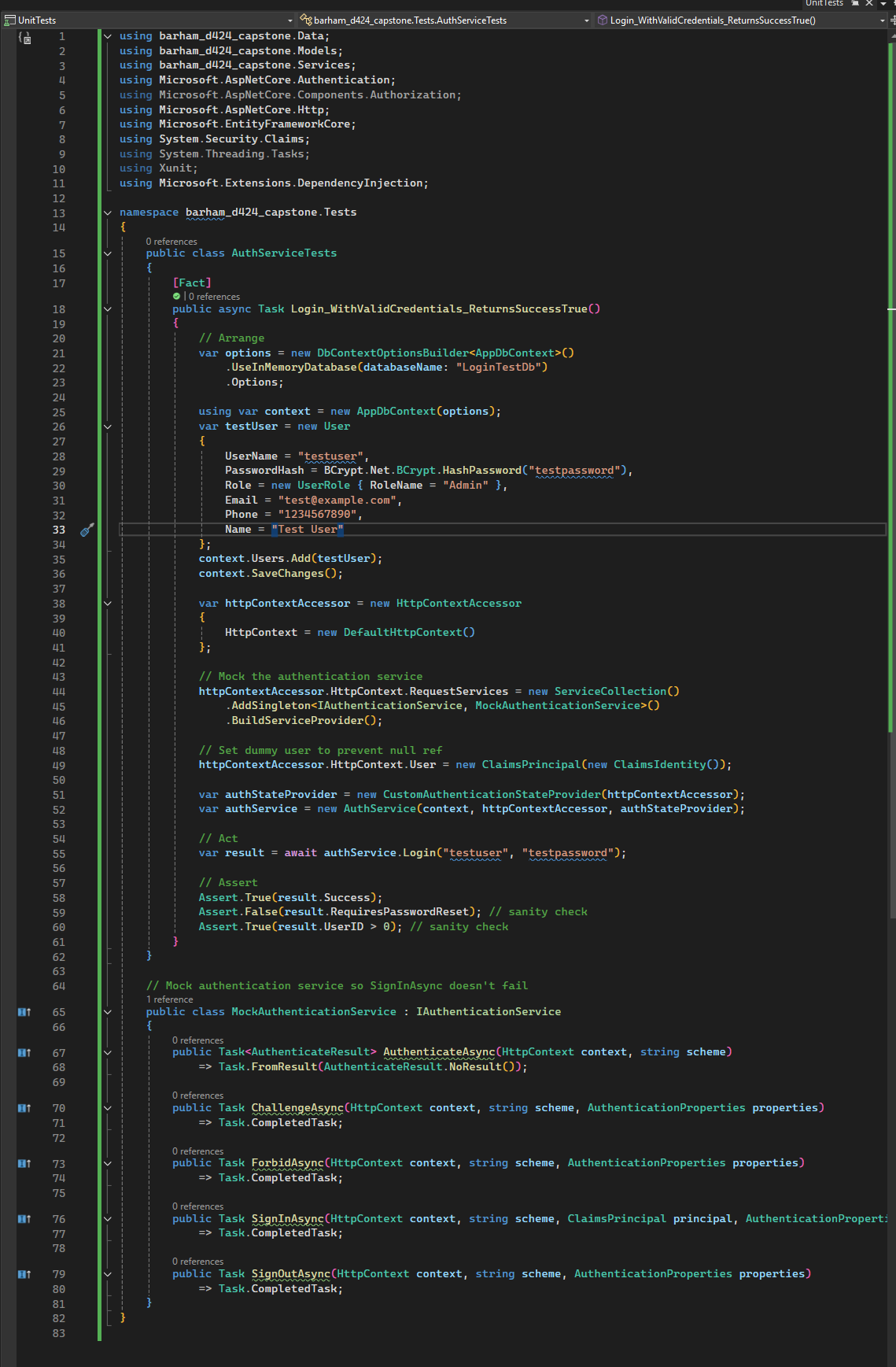
Product add functionality:

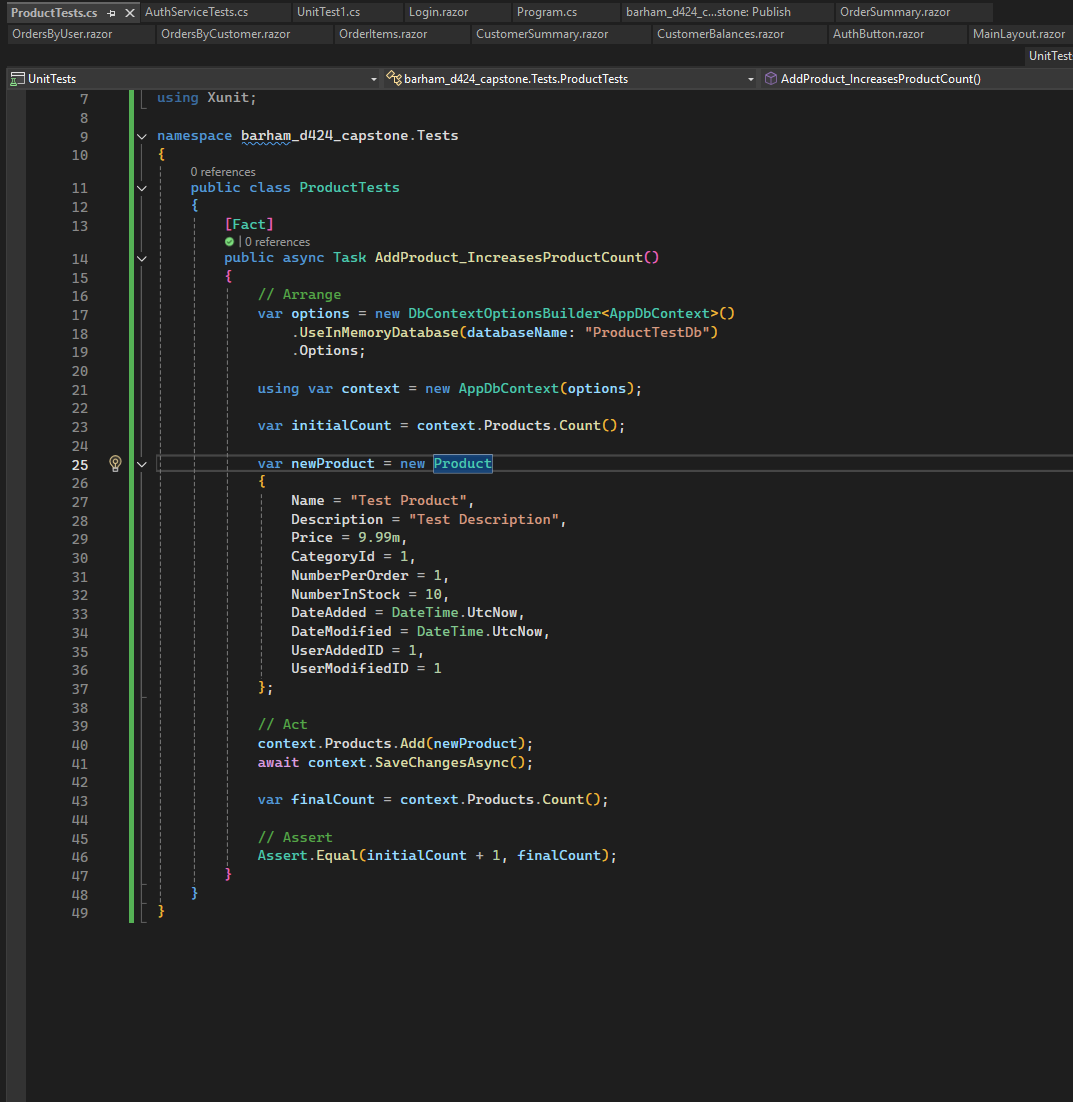


Product Added:



Example Test Code:





Link to the hosted application:

<barhamcapstone-gdbdh3evbtg2e2f2.westus-01.azurewebsites.net>

GitLab Environment link:

<https://gitlab.com/wgu-gitlab-environment/student-repos/abarha10/d424-software-engineering-capstone.git>  
  
**Set up and application launch:**

Application can be hosted on Microsoft Azure App Service. Database hosted in AzureSQL. When configuring AzureSQL, be sure to use the ADO.NET connection string and set up a non-administrator account for application connection. Additionally, the current server is set to timeout to save production costs on cloud resources. DevOps should navigate to the SQL server, settings, and turn off auto-pause in order to prevent first log-in delays.

Code can be deployed directly to production from Visual Studio by left-clicking the project and selecting “publish” while connected to the DevOps administrator account, which will allow the company to select an AppService container in the event they have multiple. Ensure the App Service container is set to the correct version of .NET (8.0, long term support), and that **web sockets** is enabled, as this allows for the server-side rendering that Blazor requires.

Both the application service and SQL database can be scaled to meet company needs, with additional costs. It is recommended after setting up admin accounts to delete test accounts and limit production account access to the software maintenance team.

**Steps:**

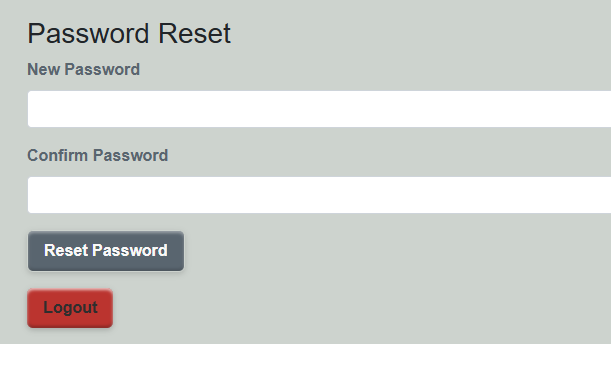
* Register an Azure account at <https://azure.microsoft.com/> and create a resource group
* Create an AzureSQL database using Create a Resource. Create a database admin login and password and store securely for DevOps
* Configure any additional SQL accounts as required by company security standards.
* Create a connection string by navigating to the SQL database and selecting Show database connection strings.
* Ensure you replace the {Your Password Here} segment of the connection string with the appropriate password when using the connection string.
* Navigate home and create a new resource, Azure App Service
* In configuration, ensure you are using .NET 8, https only, and web sockets for Blazor connectivity
* You can configure custom domains in settings to establish a domain that is easier to remember
* In Environment variables, select the connection strings tab and create a new connection string using the string from your SQL database as DefaultConnection. Remember to replace the password segment with the appropriate password.
* In Visual Studio, open the solution and left click the project. Select Publish and follow the prompts to select your appropriate App Service Container, and click the blue publish button.
* From the application, log in using an Owner level Username/Password combination to begin populating users from the User page. It is suggested to use an Owner role as only owners can create Admin users. Populate your admin users with management staff, and then allow management staff to populate the standard users from their teams.

**User Guide:**

Welcome to the new Order management application. To begin, navigate to the order management application page. You will be directed to log in using your username and password. Your initial password will be Temp(UserName)123!, replacing (UserName) with your personal username. Passwords are case-sensitive.

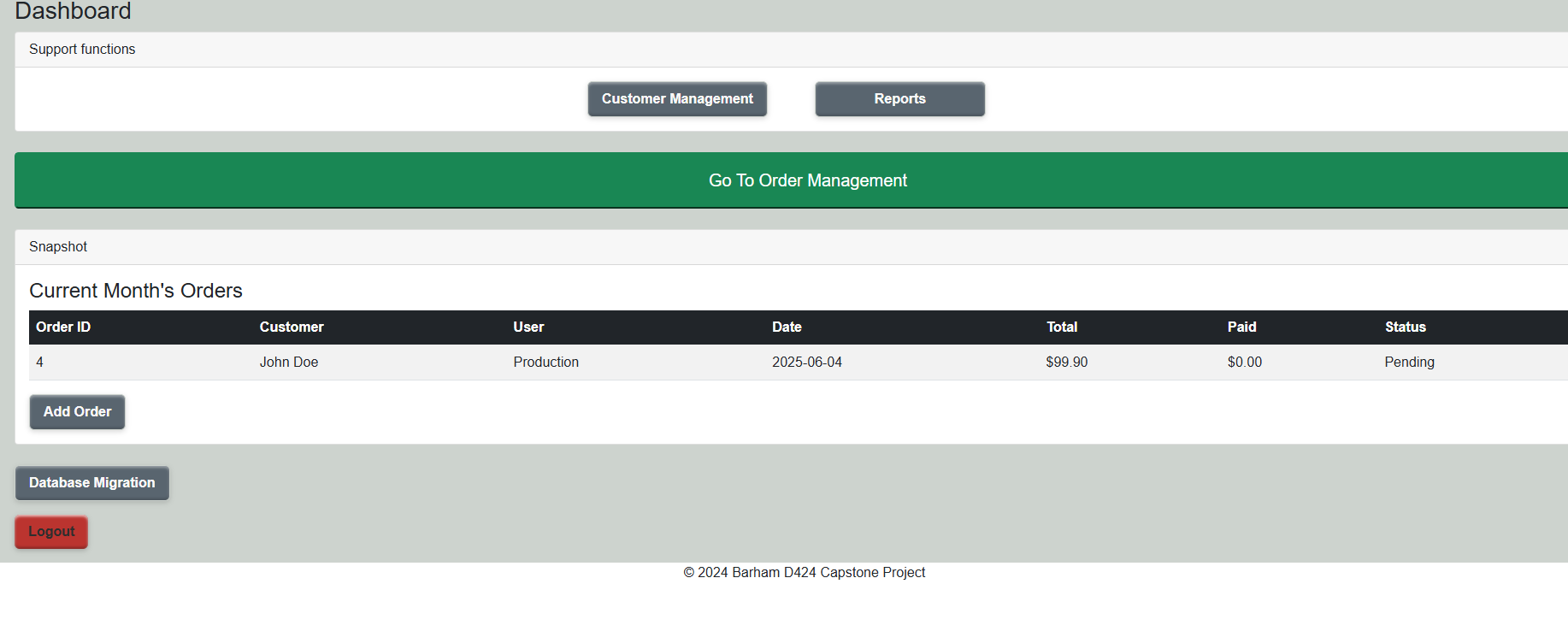


You will be directed to reset your password. Your password should contain a mix of uppercase, lowercase, numbers, and special characters. Confirm your password and you will be taken to the dashboard page. Note: you will not have to reset your password after your first login, unless you forget your password and request a password reset from your administrator.



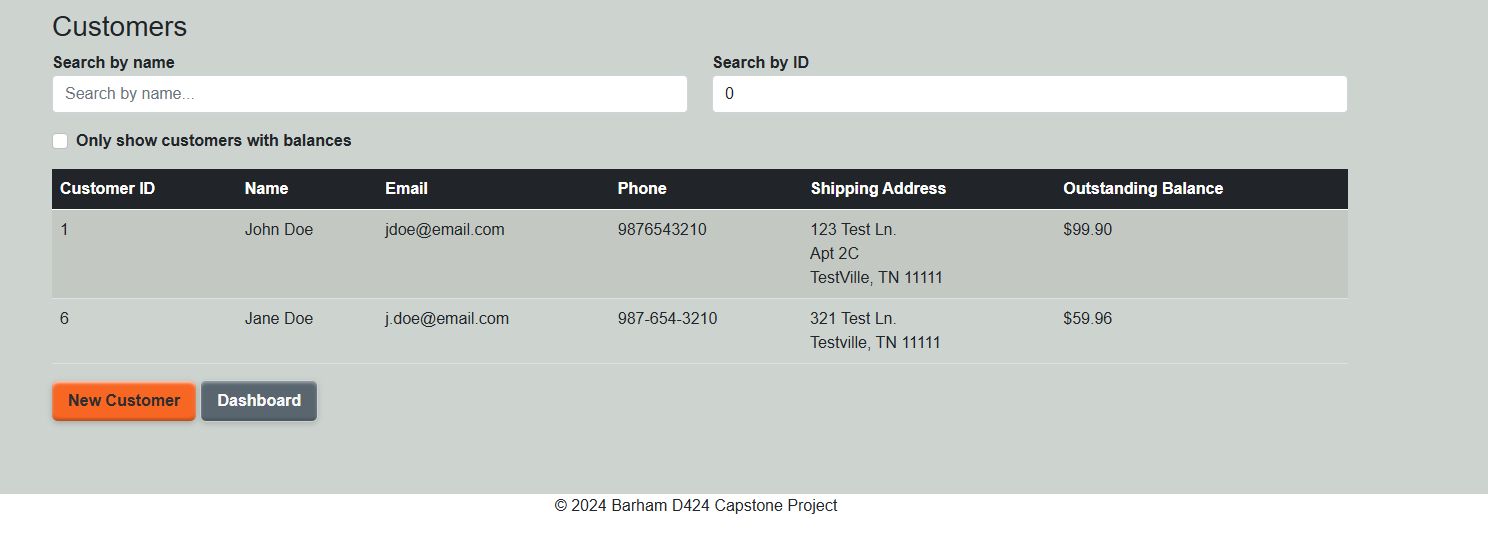
*Note: Admin and Owner level users will have a slightly different dashboard. Admin functions will be detailed below user functions.*

You will be directed to your dashboard upon logging in. The dashboard includes several important features, including the ability to manage customers, generate reports, manage orders, and migrate your existing excel order files to the new database. The dashboard also includes a snapshot of the month-to-date orders for the company.

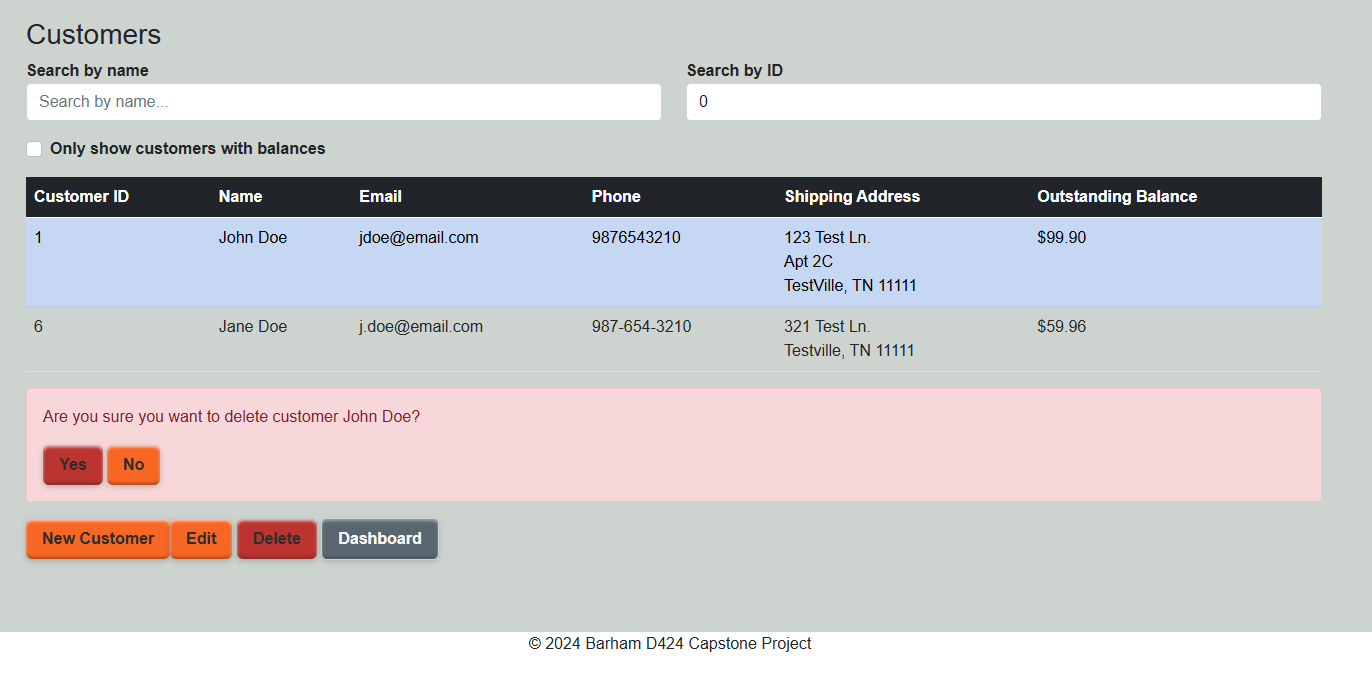


**Customer Management:**

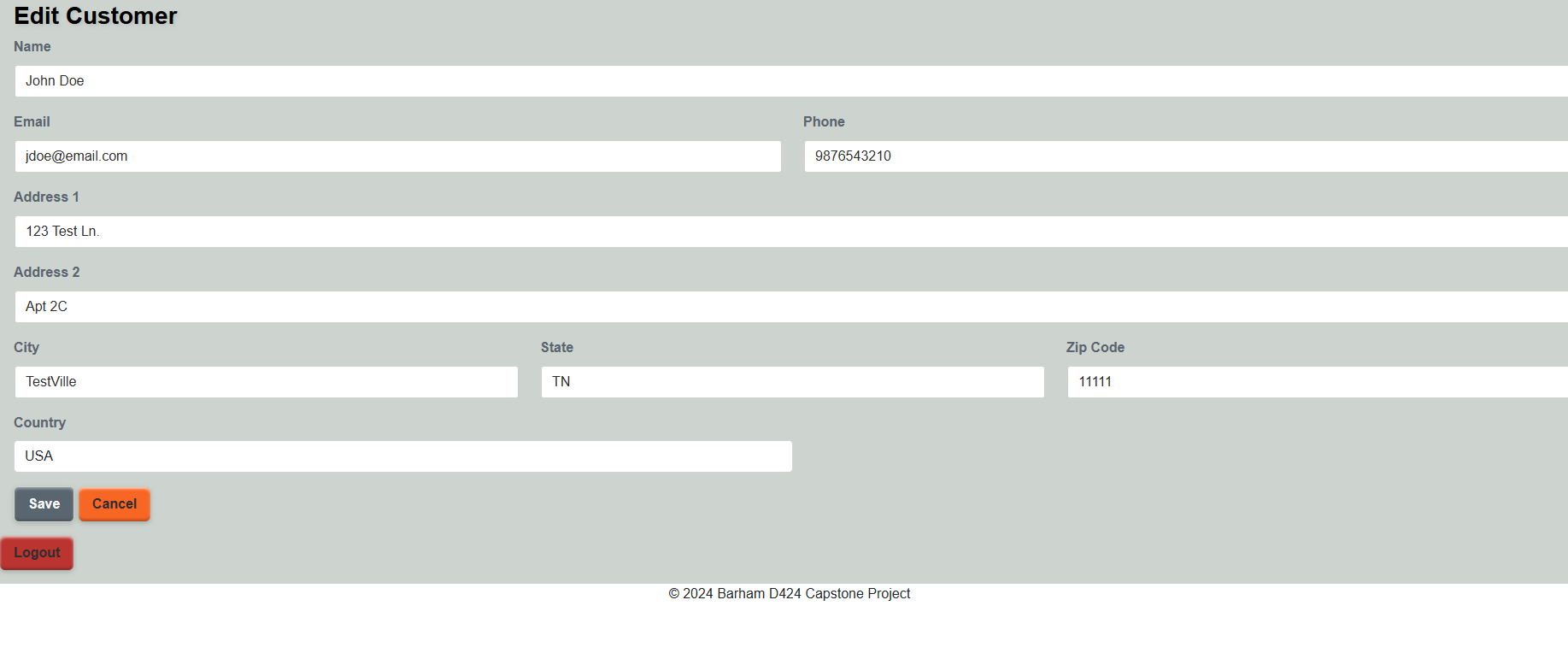
Clicking customer management will take you to the customer overview.



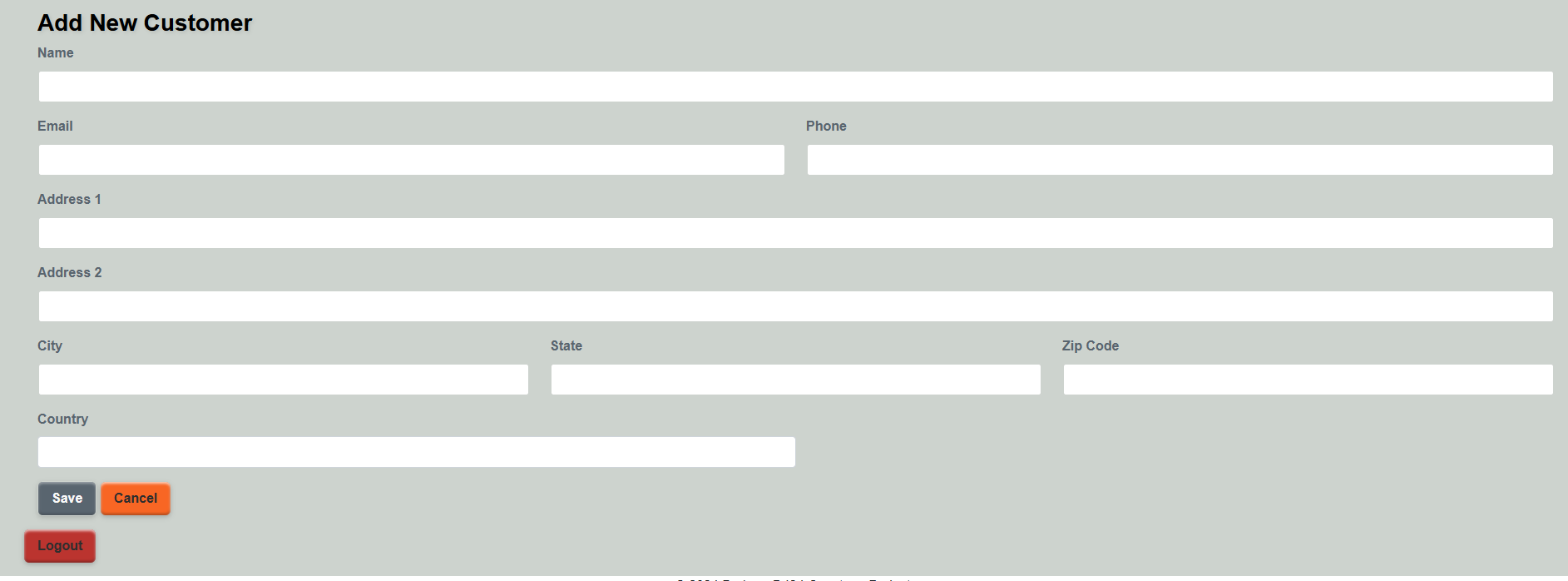
On this overview you can create new customers, delete existing customers, edit customer records, search customer records by name or customer ID, and filter customers to show only customers with balances. To delete a customer, select a customer from the list. A delete button will appear, and pressing the button will generate a delete confirmation. Selecting yes on the confirmation will delete the customer from the records.



To edit a customer record, select a customer and press the edit button. This will take you to the customer edit page. You can edit the customer’s contact information and address here. Press save to save the customer, or cancel to return to the customers page.



Select New Customer to navigate to the customer edit page with a blank customer. Saving will create a new customer, cancel to return to the customers page.



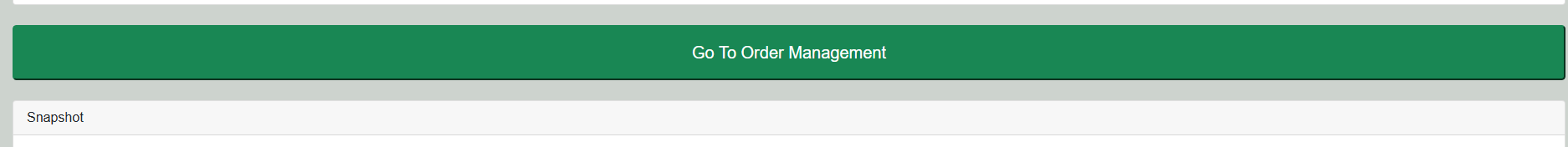
**Reports:**

From the dashboard, select the reports button. You will navigate to the reports page, where several reports are available to be viewed, including orders by customer, product sales, and customer balances. You can download a csv file of the report by selecting Download CSV. Csv files can be opened in Excel.

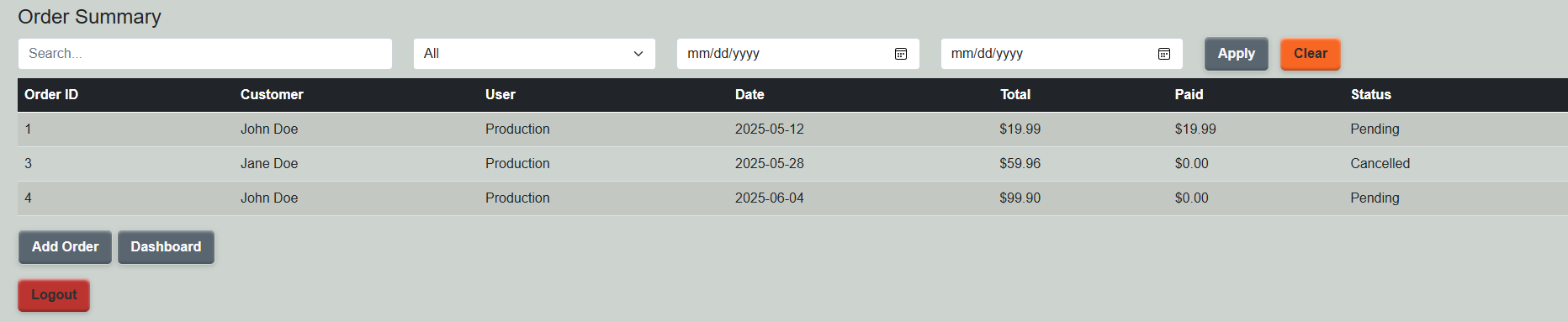


**Order Management:**

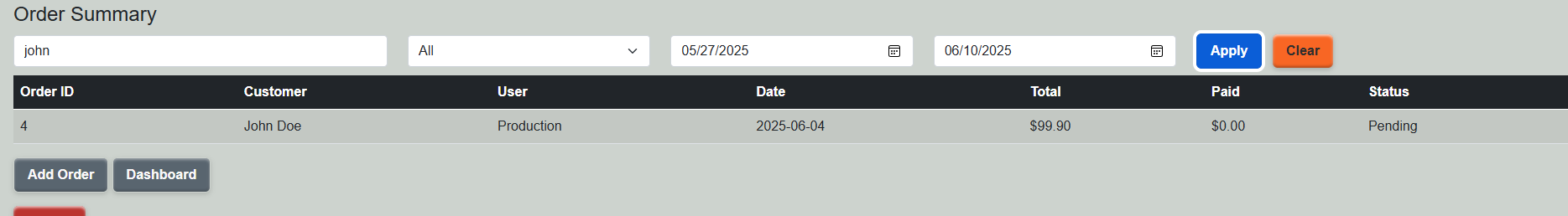
Order management can be accessed from the green Order Management button on the dashboard.



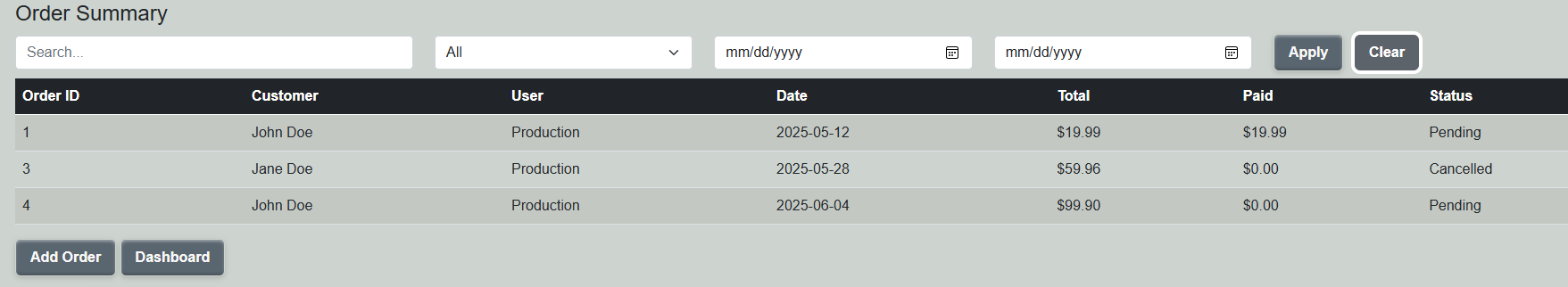
Clicking the button will take you to the Order Summary page



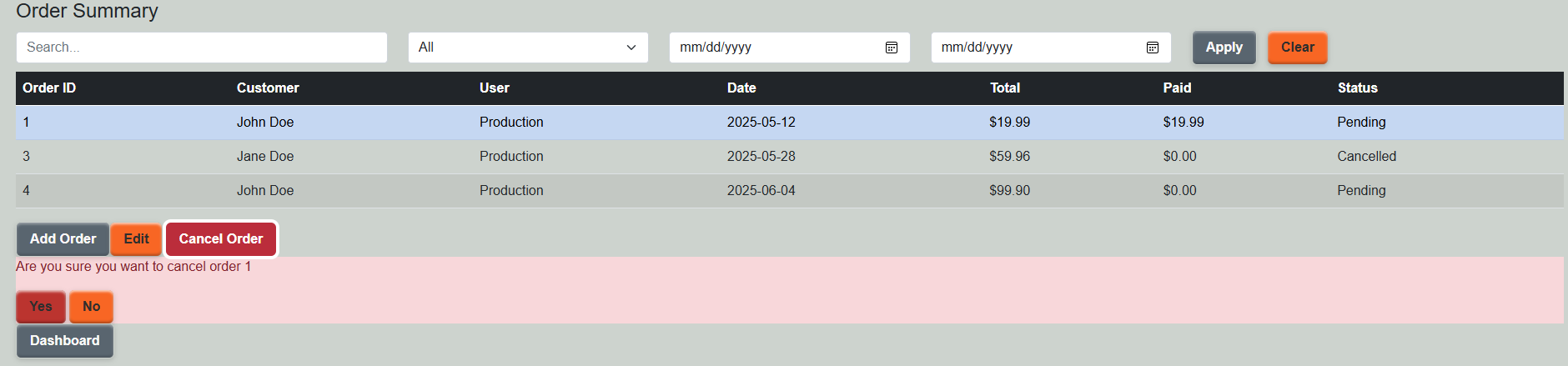
The order summary page will display a full list of orders. Orders can be filtered by date range, status, or using the search block to search by user or customer. Input the desired filter information and click “apply” to apply the filters.



To return to a full list of Orders, press the orange clear button to clear filters.

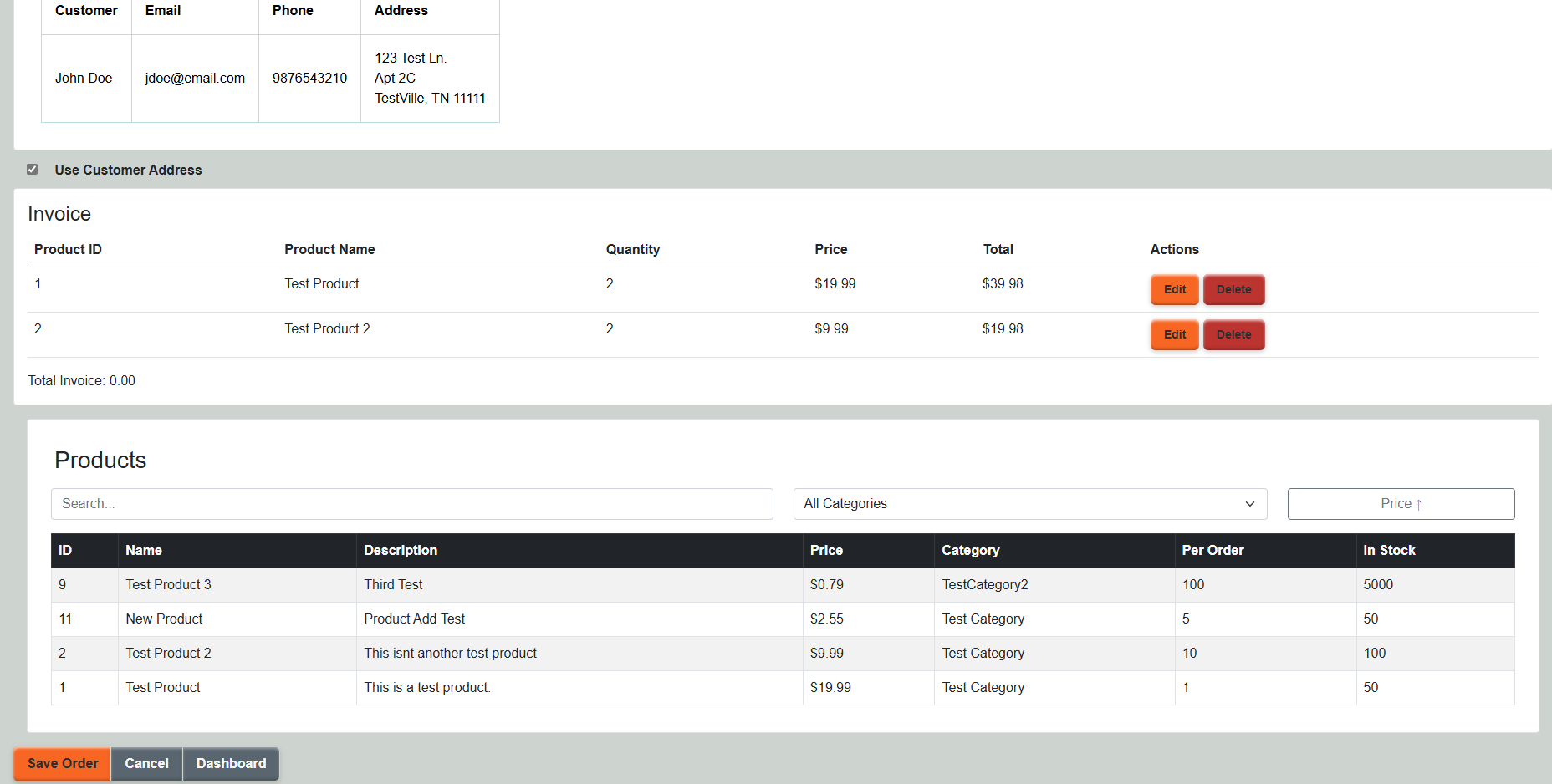


Selecting an order will give you access to the edit and cancel buttons. You can set an order’s status to cancelled by clicking the cancel order button and selecting yes in the confirmation block.



*Note: only orders that have not shipped can be cancelled.*

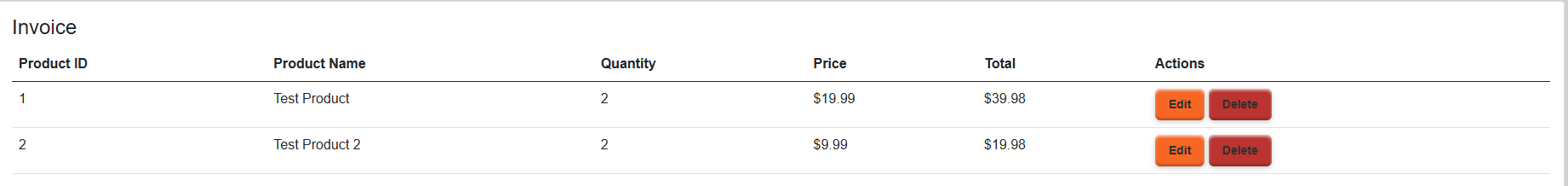
Selecting Edit will take you to the edit order page.



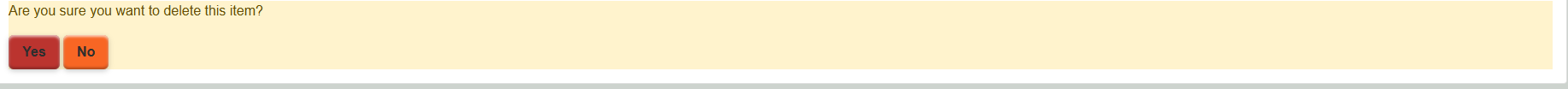
There are many elements on the edit order page. At the top is the customer information for the order. This includes the name, email and phone, as well as the shipping address for the order. If an order’s shipping address needs to be changed, you can uncheck the “Use Customer Address” checkbox to access the shipping address block and change the address the order will be shipped to.



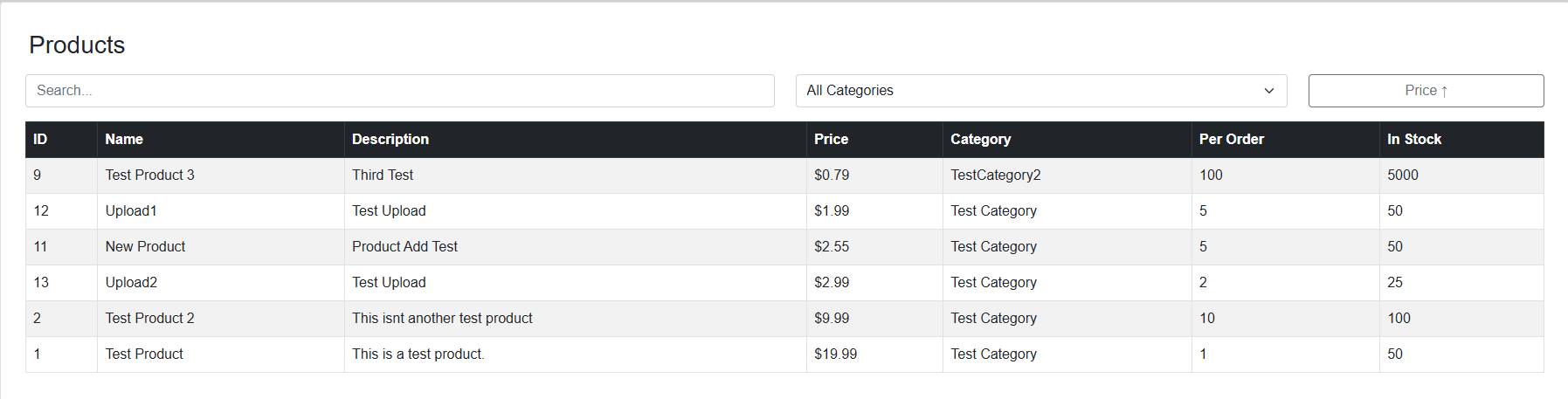
Below the address is the invoice, containing a list of the items on the order, their price, the number of each item ordered, and the total cost of the items.



Selecting Edit will allow you to change the quantity of the item being ordered, and selecting Delete will show the delete item confirmation block. Selecting yes in the confirmation block will delete the item from the order.

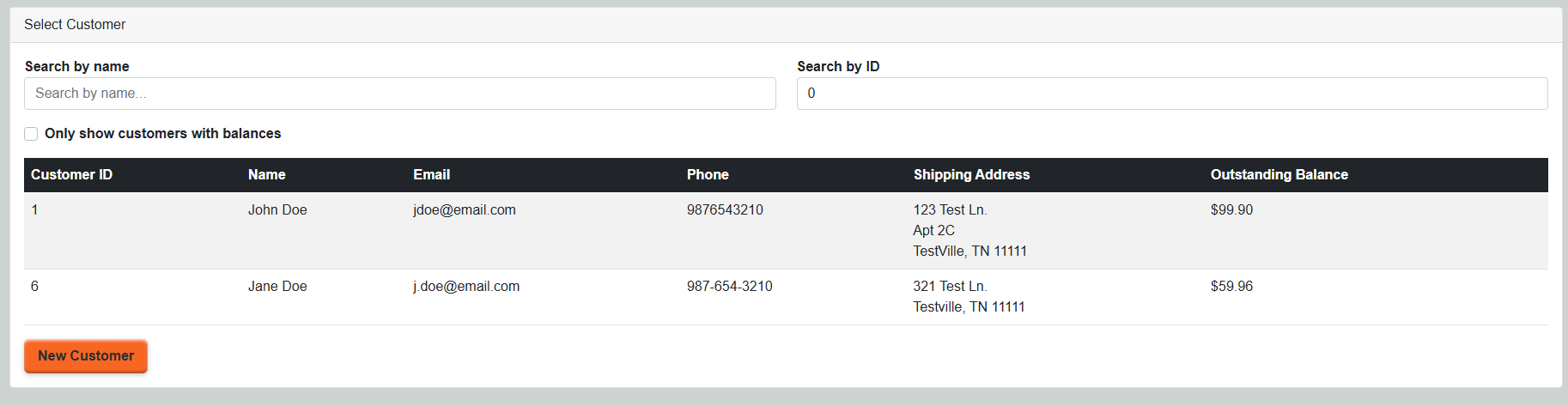


Below the invoice is a list of products.

*Note: The list of products will only display if the order can be edited. Only orders that have not shipped can be edited. If the customer needs to adjust quantities or order additional items, either execute a return or create a new order.*

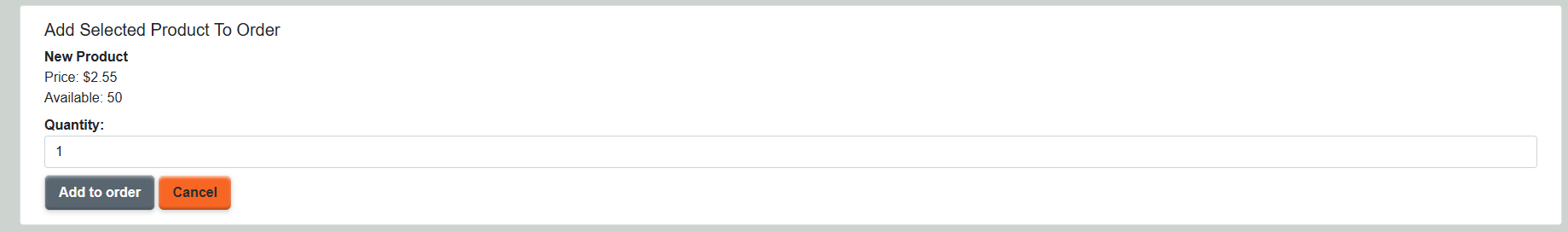
Once editing is complete, select “Save Order” to save the order, or “Cancel” to return to the orders summary without saving.

From the Orders Summary, Select Add Order to add a new order. This will take you to the order edit page. The order edit page will be the same as editing an existing order, however, instead of populating the customer at the top of the page, a list of customers will appear to select the customer that is placing the order.

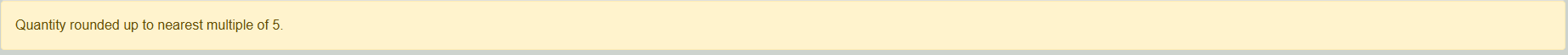


The list can be filtered to quickly find the appropriate customer, and you can easily create a new customer for the order by selecting the new customer button. To use the customer’s address on file for shipping, leave the Use Customer Address checkbox selected, or deselect the checkbox to input a custom shipping address.

To add a product to the order, select a product from the products block to show the add product block.



The add product block will allow you to specify a quantity to order before adding it to the invoice.

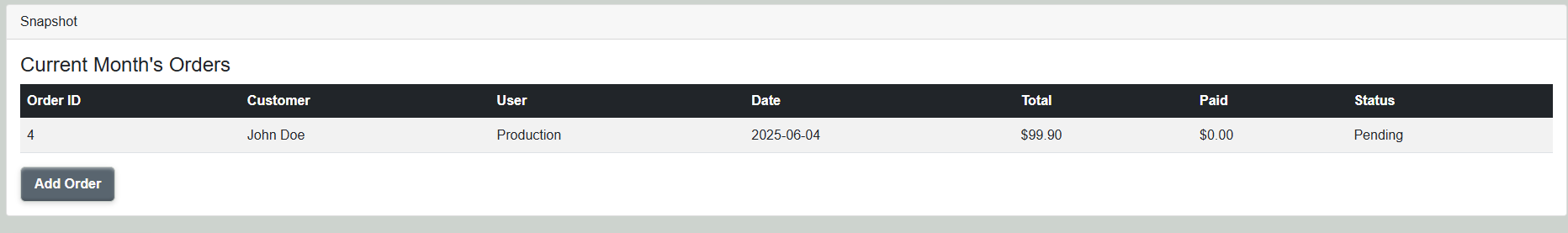


*Note: products ordered in lots will show how many products “Per Order” must be ordered. Ordering a non-multiple of this number will result in the order quantity being rounded up to the next nearest multiple of the Per Order amount.*

Select Save Order to save the order and place the order in a “Pending” status for fulfillment to begin processing. Select cancel to return to the orders summary without saving.

**Current Month Snapshot:**

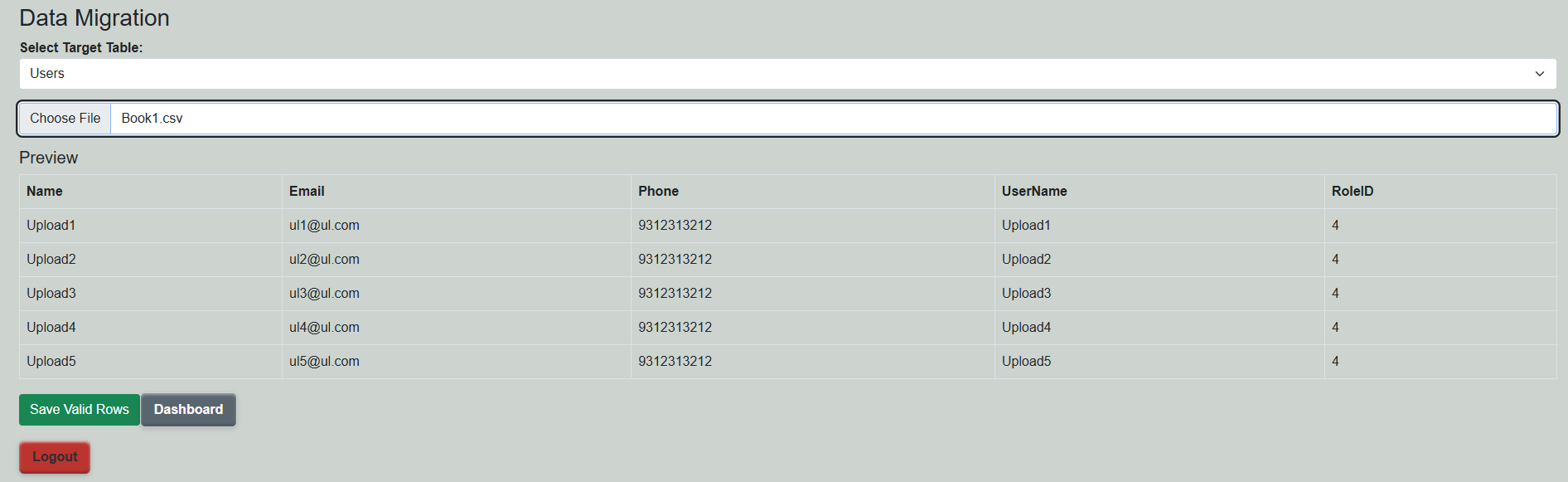
On the dashboard, the next item is the current month snapshot. This gives a high level view of the month’s orders for easy review. Additionally, a new order can easily be created from the dashboard by selecting “Add Order” from the snapshot.



**Database Migration:**

Selecting Database Migration from the dashboard will take you to the migration page, where users can upload correctly formatted CSV files for storage in the database, allowing existing excel order management tools to be integrated with the new system for tracking and archiving. From the Data Migration page, select which table you will be populating to: Users (Admin Only), Customers, or Products. Select Choose File to choose a .csv file from your computer to upload.

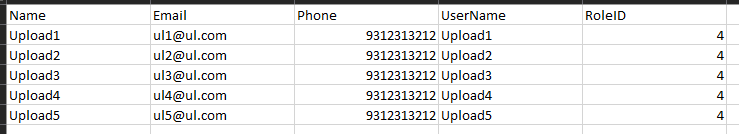
Uploading a valid and correctly formatted CSV file will show a preview of each entry you will be adding to the database.



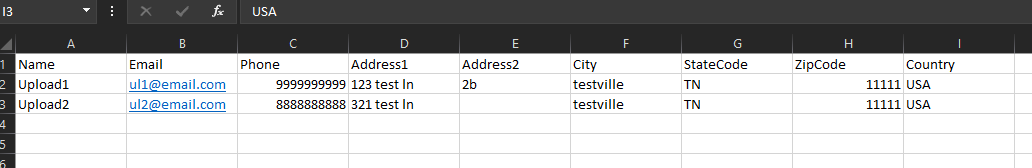
You can select Save Valid Rows to upload the csv to the database, return to the dashboard, or select a new table and file to upload.

**Valid Formats**

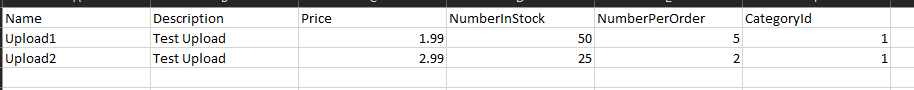
Excel files must be formatted in a specific way to successfully upload to the database. For the Users table, you must have columns with headers: Name, Email, Phone, UserName, and RoleID. Each field must be populated with a valid entry.



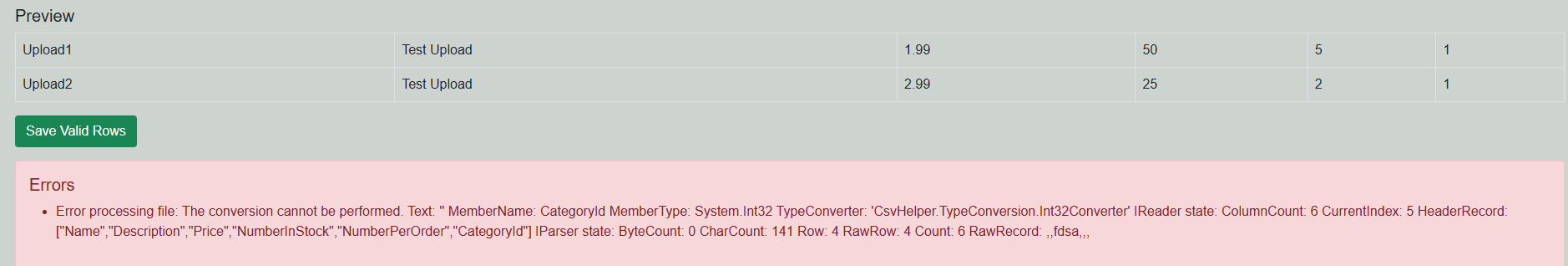
For the Customers table, each field must have a valid input except for Address2, which is only necessary for apartments or other secondary address information. Column headers are Name, Email, Phone, Address1, Address2, City, StateCode, ZipCode, and Country



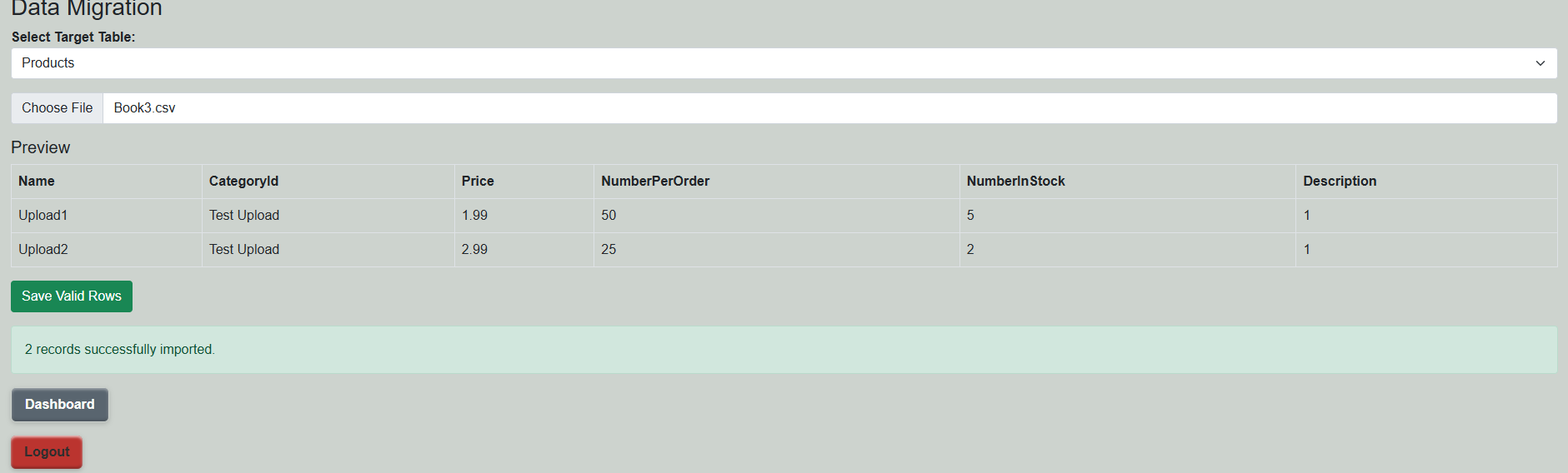
For Products, each field is required. Column headers are Name, Description, Price, NumberInStock, NumberPerOrder, and CategoryId



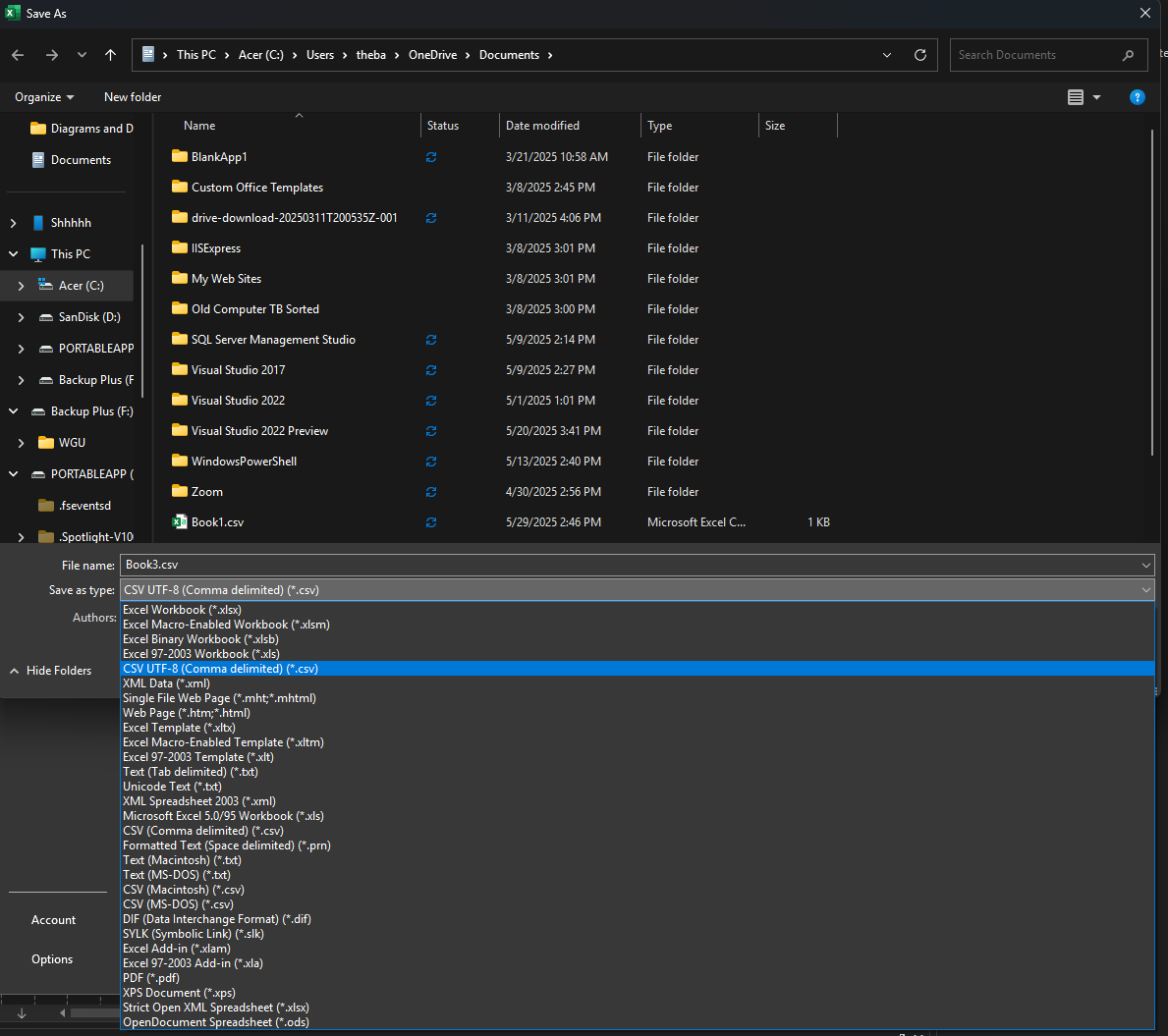
*Note: Header names must match exactly. Input fields must be valid or an error will be shown for the invalid rows.*



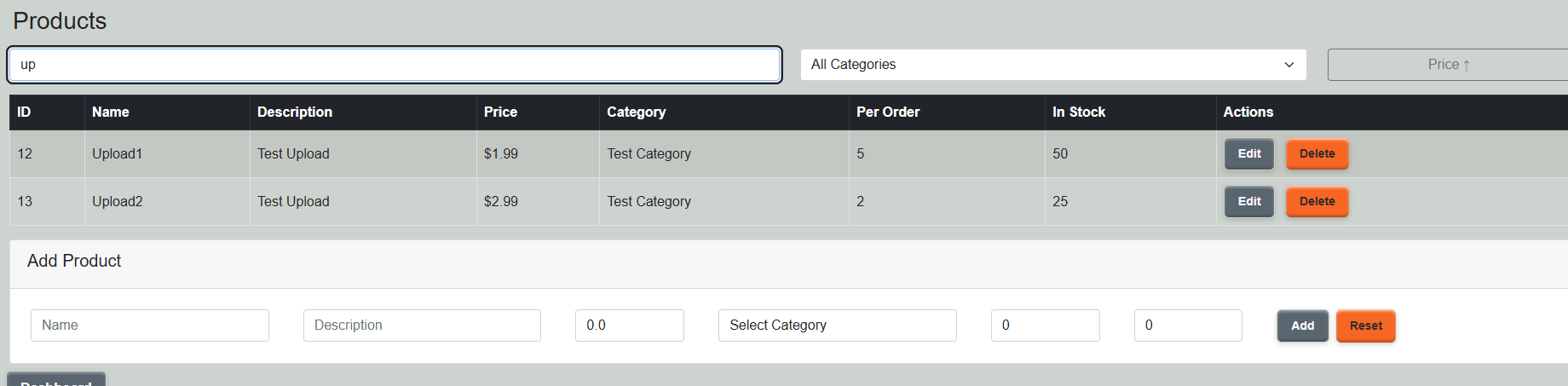
Selecting “Save Valid Rows” will save the rows to the database. A message will appear showing how many rows were imported.



To save your excel file to .csv, inside the excel file select File > Save As > Browse. Select a location to save the file, and select the save as type dropdown menu. Select “CSV UTF-8 (Comma delimited) and click save.



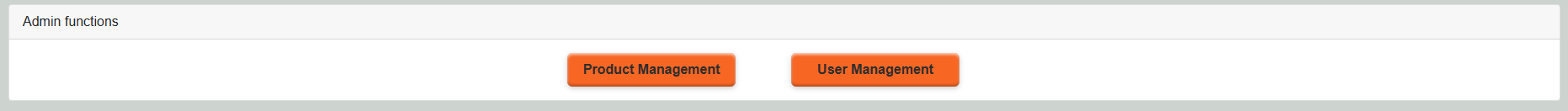
You can check your uploaded rows by navigating to the appropriate summary page.



To log out of the application, select Logout. This will return you to the login page.

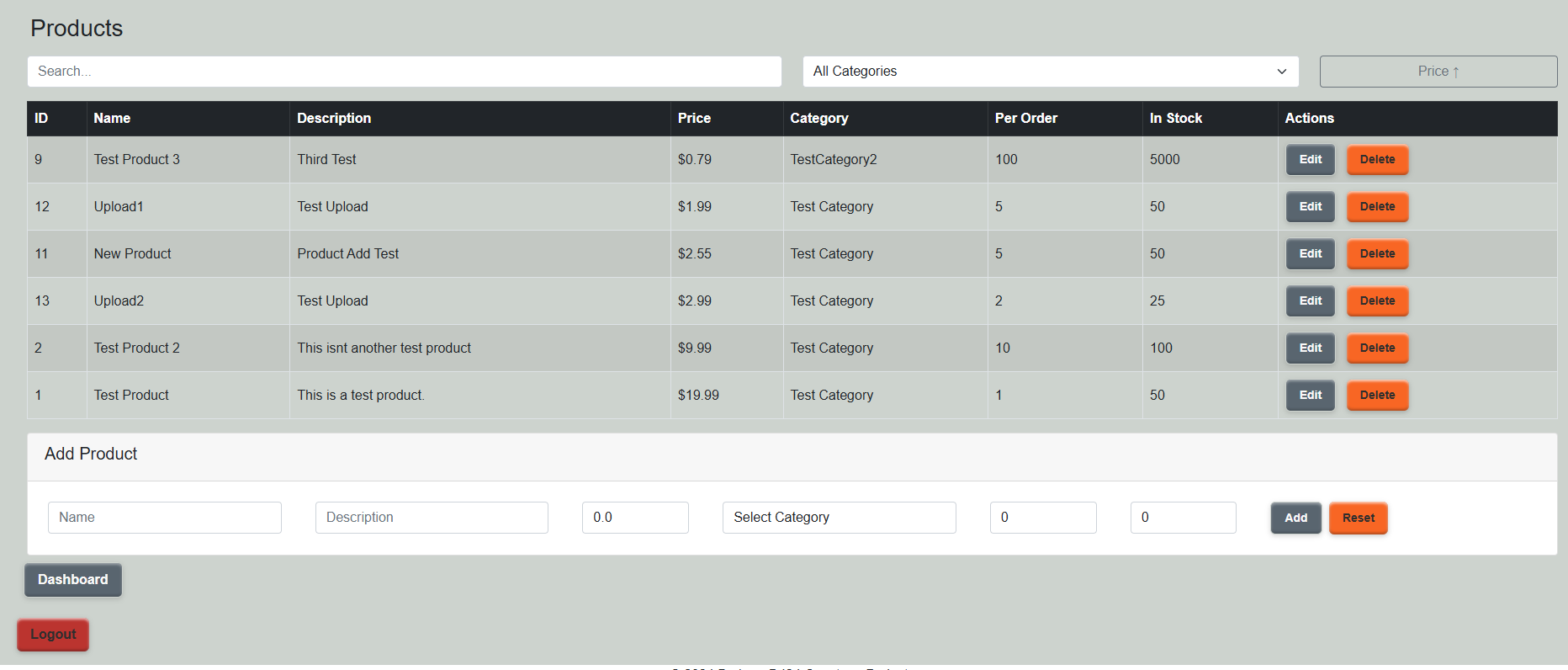
**Admin functions:**

Logging in as an admin or higher will provide additional functions to the dashboard. These functions will be denoted at the top of the dashboard with orange buttons.



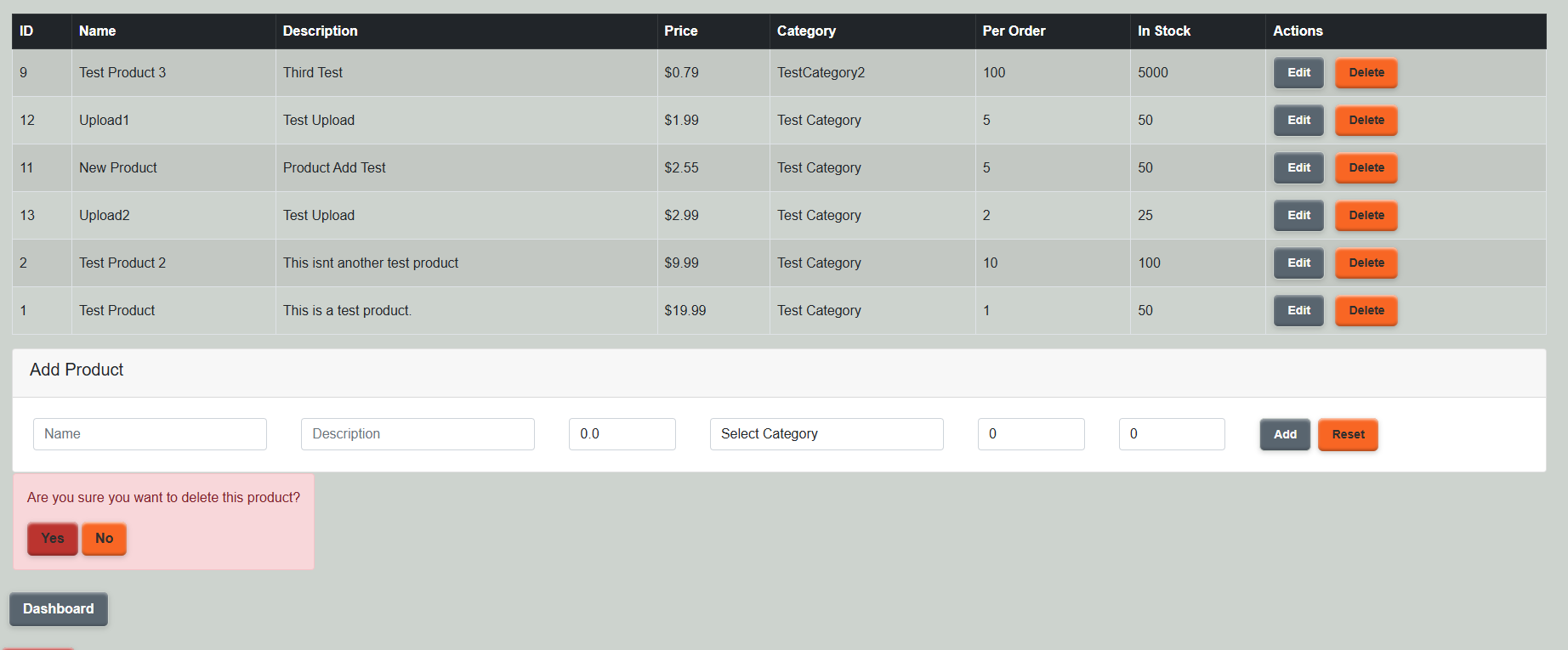
**Product Management:**

Clicking the product management button will take you to the product management screen. Here you can edit, delete and add new products.

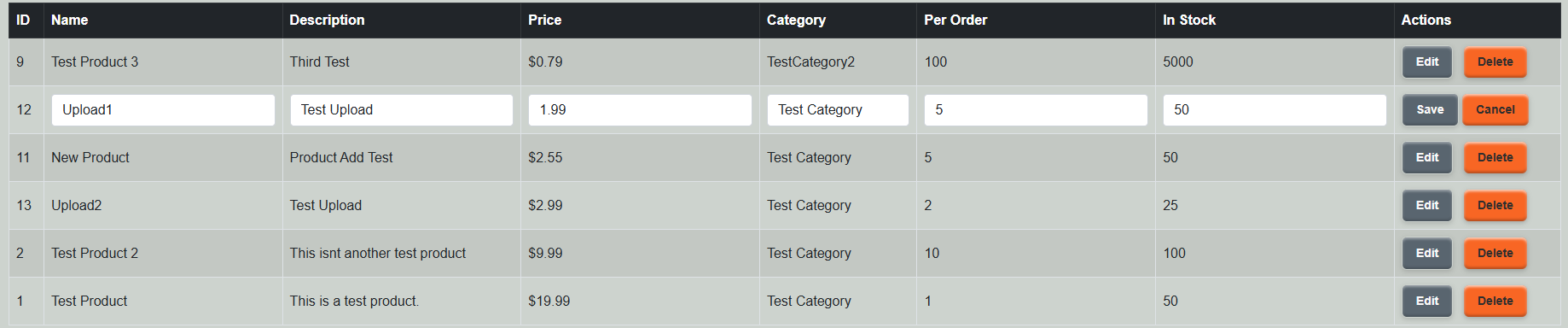


The products list can be filtered and sorted by searching name and description, or by selecting a specific category, or both. Products can be sorted by price ascending or descending using the price button.

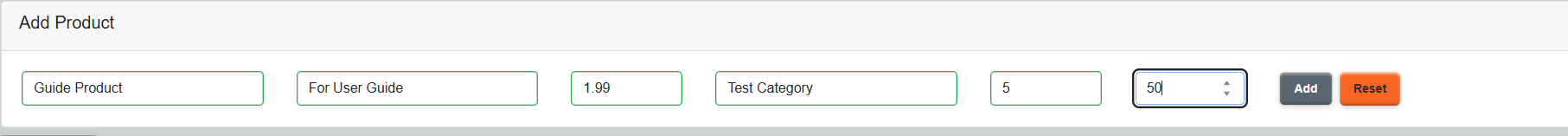
To delete a product, click the orange delete button in the product row. This will show the delete product confirmation block. Select Yes to delete the product, select No to cancel.



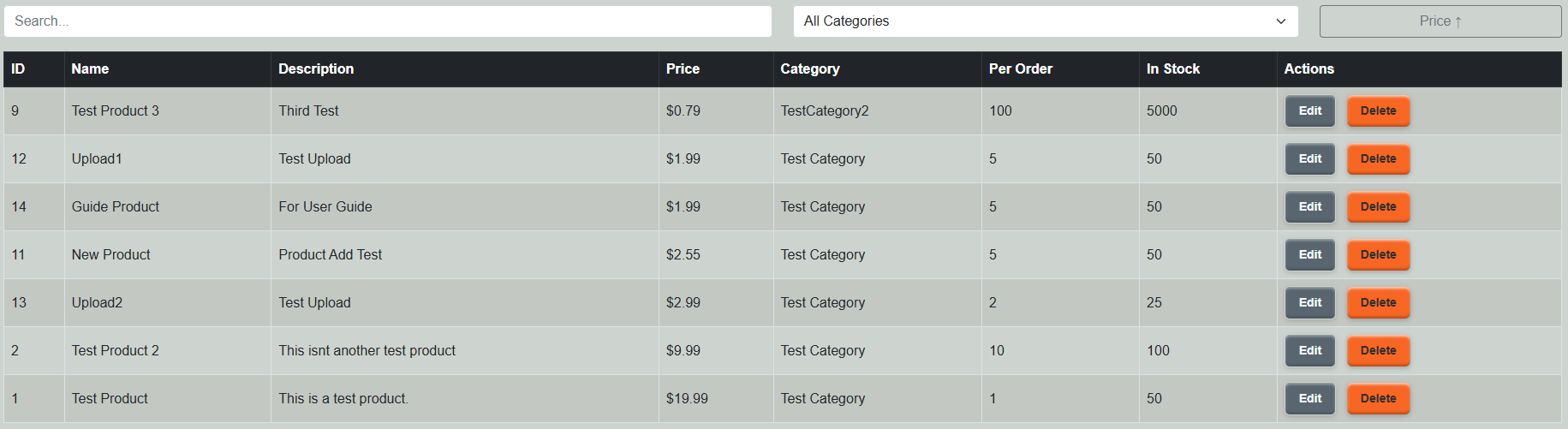
To edit a product, select the edit button. This will allow you to edit the name, description, price, category, number per order, and number in stock fields. Select Save to save the product edits, or cancel to cancel the edit.



To add a new product, use the Add Product block below. Fill the appropriate fields and click add to add the product.



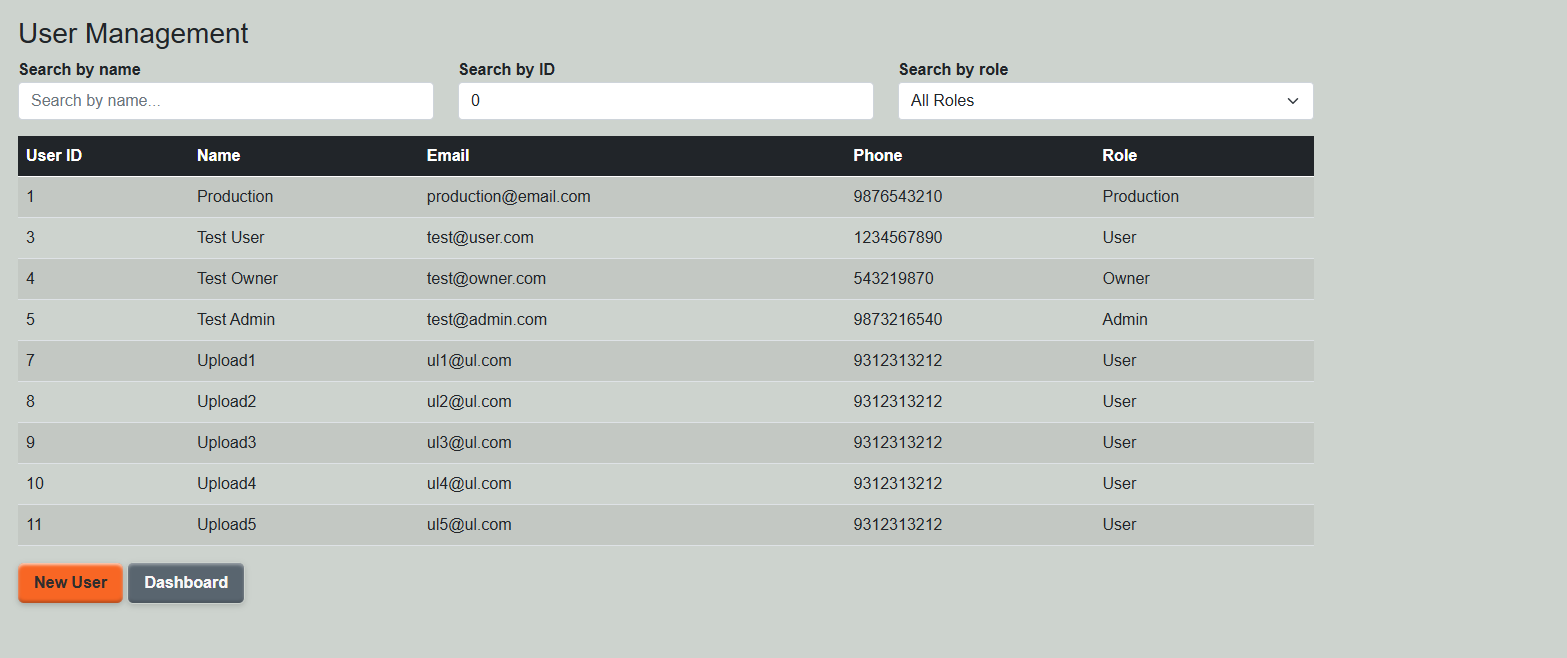
The product will appear in the products list.



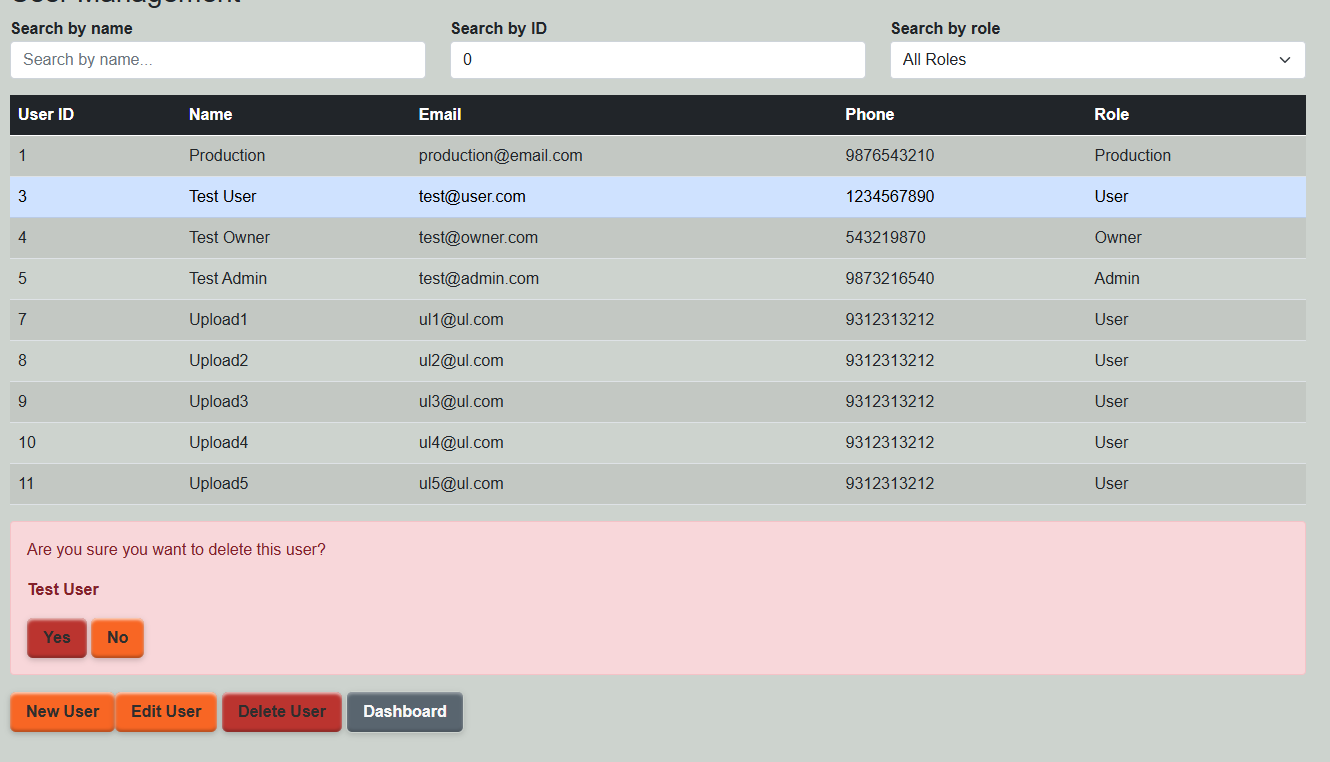
Click reset to reset the add product form.

**User Management**:

Select User Management from the dashboard to go to the user management page. Here, you will find a filterable and searchable list of existing users for the application.



Selecting a user will show the Edit and Delete buttons. To delete a user, select a user from the list and click the Delete User button. This will show the delete user confirmation block. Select Yes to delete the user, or no to cancel.

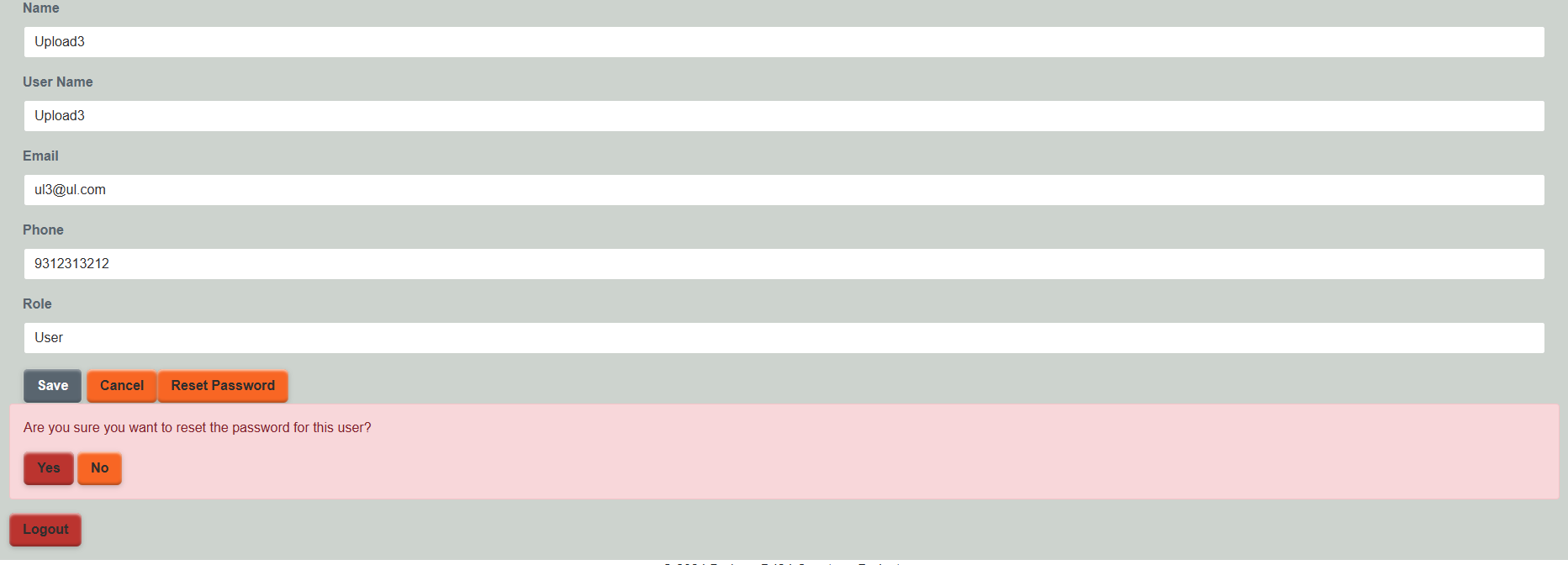


To edit the user, select the user from the list and press the Edit User button. This will take you to the user edit page.



User information can be edited on the screen. Select save to save the new information, or cancel to return to the user management page.

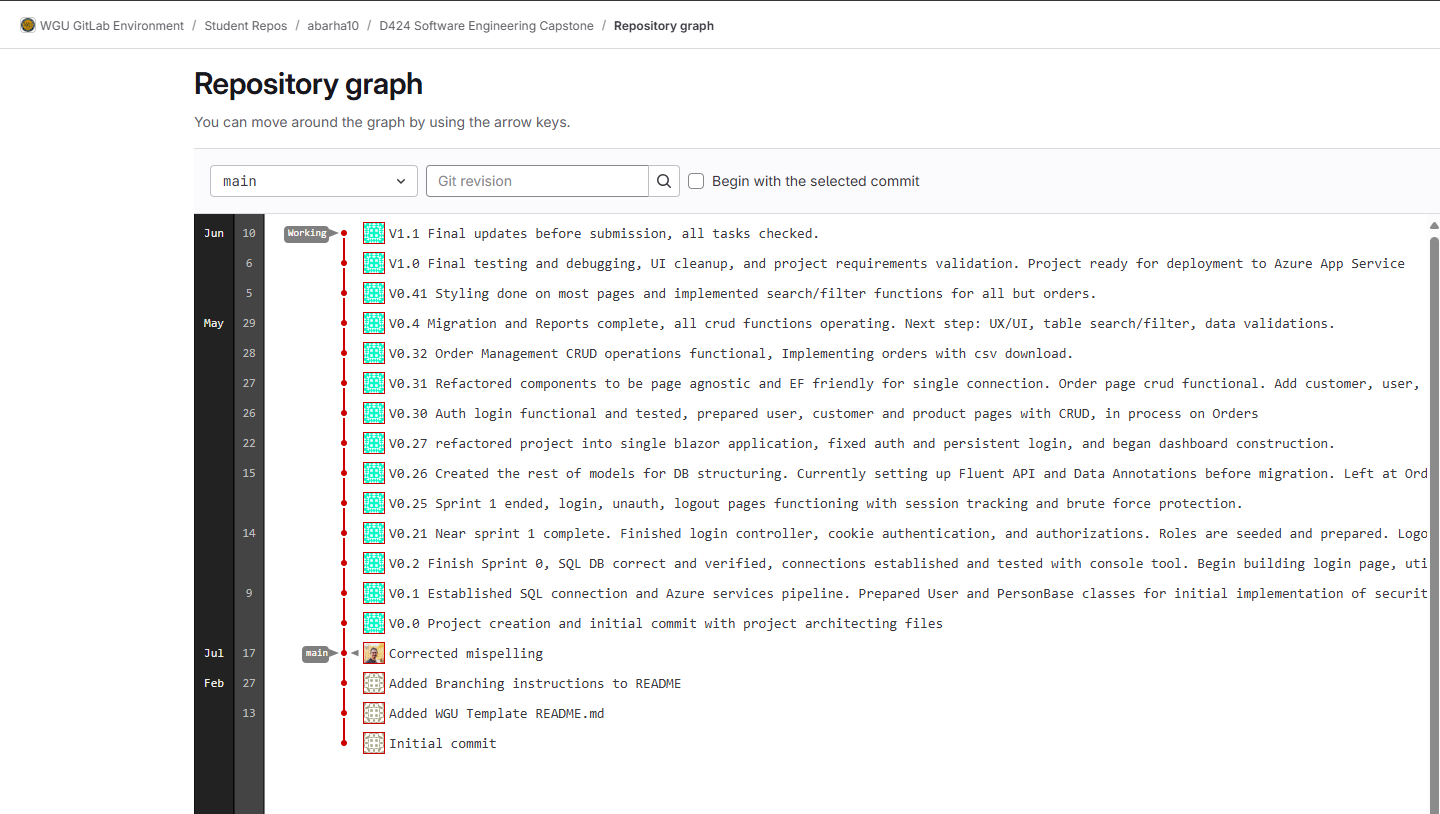
To reset a user’s password, select the reset password button. This will show the password reset confirmation block. Selecting Yes will reset the user’s password to the default password of Temp(Username)123! Replacing (Username) with the Username. It is recommended that this be done in the presence of the user, so that the user can reset their password immediately to something more secure. Selecting No will cancel.



*Note: Only users with higher level access than the user being edited will be allowed to reset passwords. Only Owner level users may elevate a user to admin roles or higher. Admin roles may ONLY create user level users.*

To create a new user, select the New User button on the user management page. This will take you to the user edit page with a blank user. Fill in the fields and click save to create the new user with the default password of Temp(Username)123! Replacing (Username) with the Username of the new user. The new user will be prompted to reset their password upon first log-in.

GitLab Branch History:



Works Cited:

No code or documentation was produced using outside resources. General .NET documentation, StackExchange, and other resources were used to troubleshoot problems during development, but all code is original.